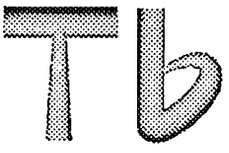
AN ELECTRONIC TOPOLOGICAL PICTUREBOOK

A THESIS
SUBMITTED TO THE DEPARTMENT OF
COMPUTER ENGINEERING AND INFORMATION SCIENCE AND
THE INSTITUTE OF GRADUATE STUDIES
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

AHMET ARSLAN

October, 1992

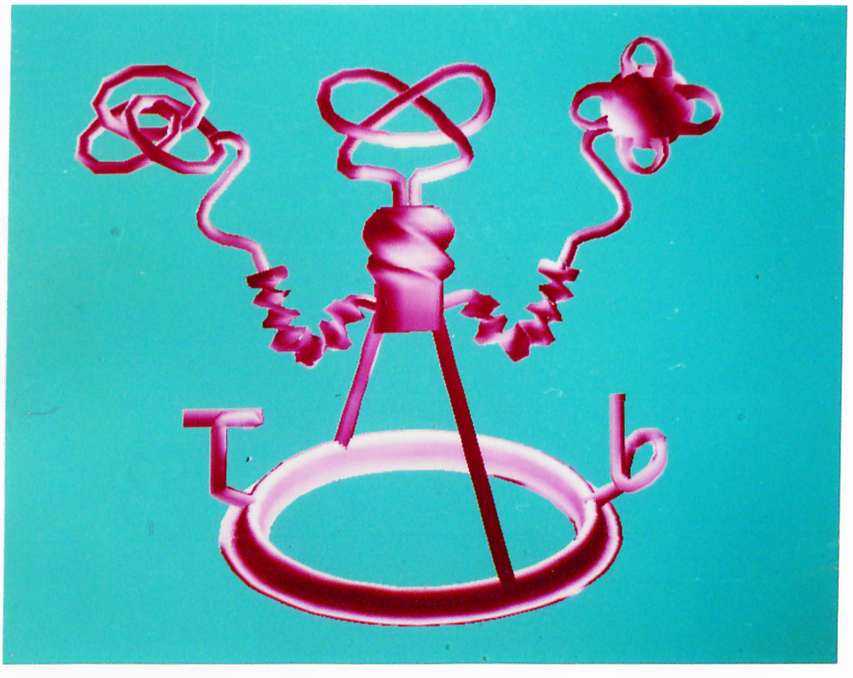
AN ELECTRONIC TOPOLOGICAL PICTUREBOOK



A THESIS
SUBMITTED TO THE DEPARTMENT OF
COMPUTER ENGINEERING AND INFORMATION SCIENCE AND
THE INSTITUTE OF GRADUATE STUDIES
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

AHMET ARSLAN

October, 1992



I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Varol Akman (Advisor)

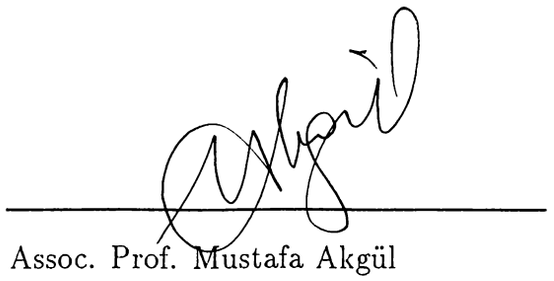
I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Bülent Özgüç

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

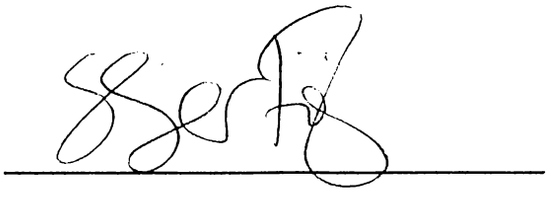
Prof. Neşe Yalabık

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



Assoc. Prof. Mustafa Akgül

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



Asst. Prof. Sinan Sertöz

Approved for the Institute of Engineering and Science:

Prof. Mehmet Baray
Director of the Institute of Engineering and Science

Abstract

AN ELECTRONIC TOPOLOGICAL PICTUREBOOK

Ahmet Arslan

Ph.D. in Computer Engineering and Information Science
Advisor: Assoc. Prof. Varol Akman
October, 1992

An electronic topological picturebook is envisaged as a computerized version of *A Topological Picturebook* by George K. Francis, Springer-Verlag, New York (1987). Francis' book is full of complicated topological figures, mostly drawn manually. The main goal of the thesis is to automate the production of such illustrations and to obtain publication-quality hardcopy using assorted techniques of computer graphics. To that end, sweeping is discussed as a major surface modeling tool. Some interactive methods are given to produce interesting topological surfaces. The program Tb (which stands for 'Topologybook') is described and various pictures generated by this software are presented. Tb is a free-form surface modeler and produces topological shapes with little effort. Central to the implementation of Tb is a paradigm of solid modeling in which computation of a shape is regarded as sweeping with some parametric variations, viz. *shape = sweep + control.*

Keywords Sweeping, surfaces, user interface systems, topology, solid modeling, B-spline curves, Bézier curves, interpolation.

Özet

ELEKTRONİK TOPOLOJİ RESİM KİTABI

Ahmet Arslan

Doktora, Bilgisayar ve Enformatik Mühendisliği
Danışman: Doç. Dr. Varol Akman
Ekim 1992

Elektronik topoloji resim kitabı, George K. Francis'in *A Topological Picturebook* [Springer-Verlag, New York (1987)] isimli kitabının bilgisayar ortamına aktarılmış hali olarak kabul edilebilir. Francis'in kitabı çoğunluğu elle çizilmiş karmaşık topolojik resimlerle doludur. Tezin genel amacı, çeşitli bilgisayarlı grafik tekniklerini kullanarak bu tür resimlerin çizimini otomatik hale getirmek ve basım kalitesinde kopyalarını üretmektir. Bu amaçla, genel bir yüzey modelleme aracı olarak süpürme yöntemi tartışılacaktır. İlginç yüzeyler elde etmek için bazı etkileşimli yöntemler verilecektir. Tb (Topologybook'un kısaltılmışı) programı açıklanacak ve bu yazılım tarafından üretilen değişik resimler sunulacaktır. Tb bir serbest-form yüzey modelleyici olup az bir çaba sarfederek çeşitli topolojik resimler çizmeyi mümkün kılar. Tb gerçekleştirimi temelde şu katı modelleme ilkesi üzerine kuruludur: Bir biçimin hesaplanması, bazı parametrik değişkenlerin kontrollu olarak süpürülmesine eşdeğerdir, yani *biçim = süpürme + kontrol.*

Anahtar Sözcükler Süpürme, yüzeyler, kullanıcı arabirim sistemleri, topoloji, katı modelleme, B-spline eğrileri, Bézier eğrileri, interpolasyon.

Acknowledgments

I would like to thank my advisor, Assoc. Prof. Varol Akman, for suggesting the problem and for continually calling my attention to new ways of looking at it.

I am grateful to Prof. Bülent Özgüç for his help with computer graphics concepts and valuable comments. Assoc. Prof. Mustafa Akgül deserves appreciation for his TEXpertise. Profs. Mehmet Baray and Erol Arkun provided the excellent research environments in which this work was performed. I would also like to thank my colleagues (especially Veysi İşler and Oğuz Işıklı) who have contributed to my research and to the preparation of this manuscript.

Asst. Prof. Mustafa Poyraz (Fırat University) kindly agreed to let me spend part of my time at Bilkent, especially during the final phases of this study.

Finally, my sincere thanks are due to my wife for her understanding and continuous moral support.

Contents

List of Figures

1 Introduction

A *free–form surface modeler* offers a set of surface design tools of which one can distinguish three types: *constructors*, *modifiers*, and *combiners* [21]. Constructors present several methods for generating 3-D models of (real or imaginary) objects. The most popular approach is to use polygons as low-level building blocks which define increasingly complex objects. For example, thousands of polygons might be used to approximate a wine glass [11]. Obviously, it is difficult and time consuming for a designer to define an object in this way. Therefore, a constructor must include higher level capabilities, such as the description of a surface by interactive positioning of the control points or by a surface-fitting operator using interpolation or approximation. One objective of this class of tools is to allow the design of desired surfaces to be performed as simply as possible. Hence, a capability to define generic constituents described by a few parameters is very useful. The constituents can be ordinary (a sphere, a torus, a box) or more complicated (a surface of revolution). Sometimes it is necessary to use a higher level component such as a B-spline (or a Bézier) surface. Such surfaces or parametric patches require numerous calculations when it comes to rendering, although they can be represented in a compact form [12, 13, 14, 23, 24, 34].

Modifiers change the shape of an existing surface. The simplest technique is to move (or add) some control points, but more advanced operations have also been developed. For example, tapering, twisting, and blending techniques are available for smoothing and deforming a surface according to continuity constraints [50]. Finally, combiners are Boolean operators applied to the surfaces designed via constructors and modifiers. Each of these three operators has proved beneficial and therefore a modeling system should provide a general, accurate, and user-friendly framework incorporating them.

In this thesis, the suggested techniques for modeling 3-D shapes are all based on sweeping. We give interactive methods for specifying and manipulating 3-D representations and discuss the desirable functionalities of an interactive graphics system to visualize the complicated shapes of low dimensional topology and knot theory [22, 37, 41, 56]. The implemented program Tb ('Topologybook') is a rudimentary graphical workbench* to help topologists (also artists, solid modelers, publishers, and geographers) to illustrate their ideas more effectively. Tb was implemented in the C programming language [38, 39] and runs on a color Sun† workstation.

Our research owes its existence to *A Topological Picturebook* [25] of George K. Francis—a book that was written to encourage topologists to illustrate their work and to help artists to understand the abstract (and invariably difficult) ideas expressed by such drawings. The reader is referred to references [2, 3, 4] for early origins of the present work (albeit from a geometrical viewpoint) and related ideas. Other relevant contributions on the theme of this thesis can be found in references [15, 48, 49].

Indulgence in a topological 'picturebook' may sound somewhat futile vis-à-vis the trend that sketching and visual presentation of topological constructions (and proofs) are slowly losing their stronghold which they once held. It may appear that the science of the 'deformation of shapes' is becoming sterile in terms of figures. However, we believe that there is definite place for a topological picturebook of the sort to be described here (also cf. [22, 37]). The following excerpt [25, p. 78] succinctly explains our view:

> "[T]he pedagogy that underlies the entire book, and which I bring out specially here, comes from Bernard Morin of Strasbourg. Pictures without formulas mislead, formulas without pictures confuse. I don't know if Bernard would say it this way, but it is how I have understood his work ... Morin's vivid, pictorial description of his bold constructions has inspired their realization in many a drawing, model, computer graphic, and film. But he insists that ultimately, pictorial descriptions should also be clothed in the analytical garb of traditional mathematics."

*The word Tb is written in this way because we hope that our Topologybook combines truth (T) with beauty (indicated by the musical notation b).

†Sun Workstation is a registered trademark of Sun Microsystems, Incorporated.

An outline of the thesis is as follows. In Chapter 2, a brief topological review will be presented. In Chapter 3, sweeping is illustrated and a mathematical model of general, profiled, depth-modulated, and twisted-profiled sweep objects is given. A new model of twisted-profiled general sweep objects with a (possibly varying) contour can also be found in this chapter. In Chapter 4, the implementation of Tb is illustrated. (The reader is referred to Appendix A for technical details of Tb software.) Here, the user interface system of Tb and modeling, editing, and local or global deformations of sweep objects are presented. In addition, ruled curves, skinning, and blending of objects are covered. Handle normalization, shading, transparent windows on objects, transformations, text and image operations, animation of sweep-based objects are also presented in Chapter 4. Finally, in Chapter 5, a conclusion is offered and some limitations of Tb (along with a description of future work) are listed.

2 Topological Review of Tb

In this chapter, we briefly compare symbolic forms of some polyhedra in topology and their curve representations in Tb. The reader will find in Section 2.1 coverage of the standard representation of 2-manifolds. Essentially, the transformation of a surface from a symbolic form to a visual representation is where we use sweeping as a modeling technique. This point will be made clearer in Section 2.2.

2.1 Normal Forms for Surfaces

Suppose we are given a surface made of (say) rubber. If we distort it in a continuous manner without tearing it, the resulting surface will have the same chromatic number[‡]. Similarly, if a graph (which we may imagine to be made of wire) is distorted without tearing, many of the properties of the graph remain unchanged [27, 53, 56]. The study of those properties of a figure which do not change after deformation is a main concern of topology. It is useful to give a precise mathematical definition of the kind of deformation we have in mind, viz. a homeomorphism.

Let E and F be point sets in Euclidean 3-space. Assume that for each point $x \in E$ there exists a corresponding point $f(x) \in F$ and for each point $y \in F$ there exists exactly one point $x \in E$ such that $f(x) = y$. Such a mapping f from E to F is called a *1-1 correspondence*. The inverse transformation which leads from each point $y \in F$ back to the original point $x \in E$ is denoted by

[‡]If S is a surface and the countries of each map on S are colorable with n colors, while not every map on S is colorable with $n - 1$ colors, then we say that n is the *chromatic number* of S.

f^{-1}, and we write $f^{-1}(y) = x$. A transformation f of E onto F is defined to be *continuous* at a point x_o of E, if for each $\varepsilon > 0$ there exists a $\delta > 0$ such that for every $x \in E$ with a distance less than δ from x_o, the distance between $f(x)$ and $f(x_o)$ in F is less than ε. If f is continuous at each point of E, we say f is a *continuous transformation* of E onto F. If f is a 1-1 correspondence of E onto F and f and f^{-1} are both continuous, then f is called a *homeomorphism.* If there exists a homeomorphism of E onto F, then E and F are said to be *homeomorphic* and F is said to be the *topological image* of E. Some examples of pairs of homeomorphic figures are as follows:

- A circular disk, a square.
- The surface of a sphere, the surface of a tetrahedron.
- The interior of a circle, the interior of a rectangle.
- A set of n distinct points, another set of n distinct points.
- The surface of a torus, the surface of a tea cup.
- A solid torus, a tea cup.

A *circular disk* is the surface (the interior and the circumference) of a circle. Each topological image (in 3-space) of a circular disk is called a *2-cell.* Similarly, the topological image of an edge is called a *1-cell.* Consequently, a *0-cell* is just a point. By the term 'polygon' (triangle, quadrilateral, pentagon, hexagon, etc.) we do not restrict ourselves to plane figures. We define a *polygon* with r sides (r-gon) as a 2-cell which has its circumference divided into r arcs by r vertices. The arcs are the *sides* of the polygon.

Given a finite number of polygons, let the total number of all of the sides of the polygons be even. These sides are given in pairs and are labeled. Two sides of a pair are labeled with the same letter. It is furthermore supposed that for each side an orientation is given (say, indicated by an arrow). Identifying each pair of sides such that the arrows coincide (i.e., point in the same direction) if one obtains a connected pseudograph consisting of the vertices and the edges (superimposed sides), then the figure so obtained is a *polyhedron.* The *faces* of the polyhedron are defined as usual.

The *plane representation* of a polyhedron is all the information that have been mentioned so far: the polygons, the pairing of the sides, and the orientation of the sides (before identification). A symbolic description of a polyhedron is created in the following way. For each polygon in the plane representation, choose an orientation. Then, for each polygon, write down the cyclic order of the sides as a row of the corresponding labels (put a minus sign after the label if the orientation of the polyhedron is opposite to the orientation of the side). As a result a *scheme* (denoted by Σ)

$$a \quad b \quad c^{-1}$$

is obtained with the following properties: (i) Each letter appears exactly twice in Σ, and (ii) The set of the rows in Σ cannot be separated into two disjoint subsets such that property (i) holds for each of these subsets. The following operations change Σ (but not the polyhedron it represents): (i) Put the first letter of a row behind the last letter, (ii) Whenever a certain label appears, say a, replace it with a^{-1} (and a^{-1} with a), and (iii) simultaneously change all the signs in a row and reverse the cyclic order.

To get a general idea about all possible kinds of polyhedra one proceeds as follows. Given a polyhedron P and its symbolic representation scheme Σ, one defines four *elementary operations*. The property of these operations is that although they change P as well as Σ, the surface itself is not changed. The elementary operations are as follows:

- SU1 (Subdivision of dimension one): An edge of the polyhedron is divided into two new edges by taking an inner point of the edge as an additional vertex.
- CO1 (Composition of dimension one): Inverse of SU1.
- SU2 (Subdivision of dimension two): Two vertices of a polygon in the polyhedron are connected by a new edge dividing the polygon into two new polygons.
- CO2 (Composition of dimension two): Inverse of SU2.

Two polyhedra P and P' are said to be *elementarily related* if P can be transformed into P' by using a finite number of SU1, CO1, SU2, and CO2. A polyhedron is *orientable* if one can choose an orientation for each polygon such that each edge is used in both possible directions. The alternating sum $E(P) = \alpha_0 - \alpha_1 + \alpha_2$ for a polyhedron P is called its *Euler characteristic*. Here α_0, α_1, and α_2 denote the number of vertices, edges, and polygons, respectively. (In Σ, α_2 = the number of rows and α_1 = the number of different letters.)

It turns out, by the fundamental theorem of the classification of 2-manifolds [53], that each polyhedron is elementarily related to one of the following normal forms:

$$\begin{array}{ll} (H_0) & a\ a^{-1} \\ (H_p) & a_1\ b_1\ a_1^{-1}\ b_1^{-1}\ \ldots\ a_p\ b_p\ a_p^{-1}\ b_p^{-1} \\ (C_q) & c_1\ c_1\ c_2\ c_2\ \ldots\ c_q\ c_q \end{array}$$

H_0 is the normal form for a sphere (Euler characteristic = 2). H_p is the normal form for a p-tori, $p = 1, 2, \ldots$ (Euler characteristic = $2 - 2p$). Finally, C_q is the normal form for a nonorientable surface (Euler characteristic = $2 - q$, $q = 1, 2, \ldots$).

2.2 Simple Topological Shapes in $\mathbb{T}^2$

Consider a rectangle and assign a definite sense of direction (orientation) to the perimeter. Denote the sides by a, b, a, b in that order. Place an arrow on each side such that the two sides a (or b) are made to coincide and such that the heads of the arrows coincide as well. The "row"

$$aba^{-1}b^{-1}$$

is considered as a symbolic description of a polyhedron (e.g., a torus). It represents the four sides of the rectangle. The notation a^{-1} in above formula means that the third side is denoted by a but its arrow points opposite to the given orientation of the rectangle. As another example, consider a *lune*, i.e., a polygon with only two sides. Put arrows on the two sides such that the symbolic row reads

$$aa^{-1}$$

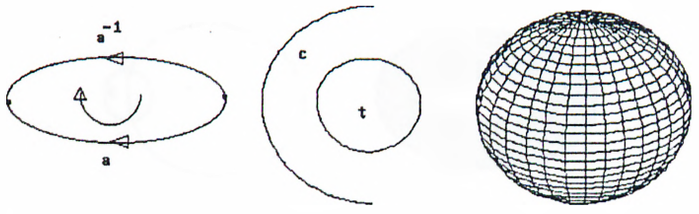


Figure 1 The representation of a sphere as a 2-gon aa^{-1} in topology. A sphere can be represented with two curves (a contour c and a trajectory t) in $\top\flat$.

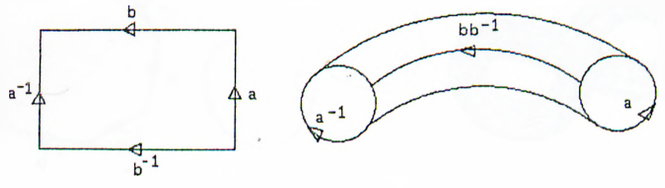


Figure 2 A rectangle represented by $aba^{-1}b^{-1}$ is transformed into a torus via an appropriate identification of the sides.

This defines a division of the sphere into just one lune. That is, a sphere can be represented in topology as aa^{-1} and this is equivalent to a 2-gon (cf. Figure 1). A sphere can be represented in $\top\flat$ by the help of two curves (a contour curve c and a trajectory curve t).

A torus can be denoted $aba^{-1}b^{-1}$ as a symbolic representation and with a rectangle as a topological representation. Figure 2 shows the representation of the torus as a rectangle and prescribes how it can be constructed by the help of the rectangle. Figure 3 shows the representation of the torus by the help of the contour and the trajectory curves. Figure 4 depicts the representations of 2-tori in topology and $\top\flat$. Symbolic representation of a 2-tori[§] is $aba^{-1}b^{-1}cdc^{-1}d^{-1}$.

[§]Of course, it is difficult to obtain an n-tori in this way. In Section 4.2.15, we show how an n-tori (viz. a sphere with n handles) is obtained by the help of its symbolic representation.

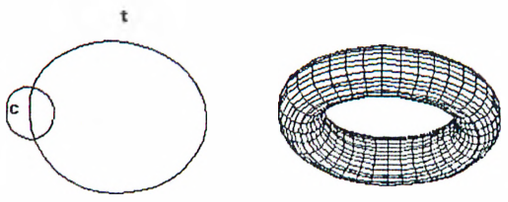


Figure 3 A torus can be defined by a contour curve c and a trajectory curve t in $\top\flat$.

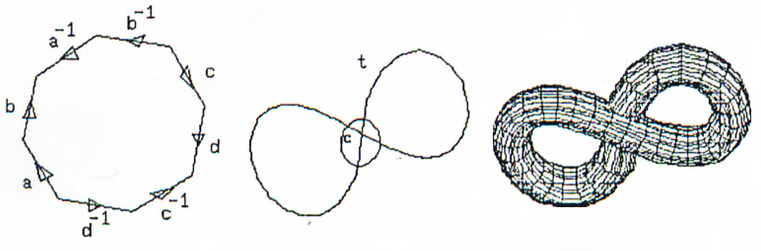


Figure 4 A 2-tori is represented as $aba^{-1}b^{-1}cdc^{-1}d^{-1}$ and can be defined by the appropriate c and t curves in $\top\flat$.

There are also polyhedra which cannot be embedded in 3-space without penetrating themselves. Consider a rectangle with the symbolic row

$$baba^{-1}$$

as in Figure 5a . If we first make the identification along the sides labeled b, we get a *Möbius strip* [53]. Figure 5b shows the curve representation of a Möbius strip in $\top\flat$. The contour curve c is twisted by the twist curve tw while it is swept along the trajectory curve t.

If we first identify the sides labeled a in

$$aba^{-1}b$$

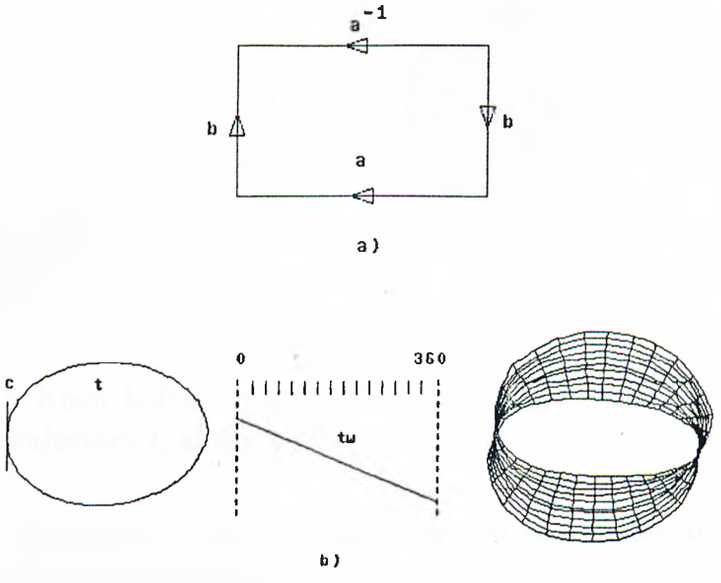


Figure 5 A Möbius strip is defined by a rectangle with $baba^{-1}$ and by a contour c, a trajectory t, and a twist curve tw in $\mathsf{T}\flat$.

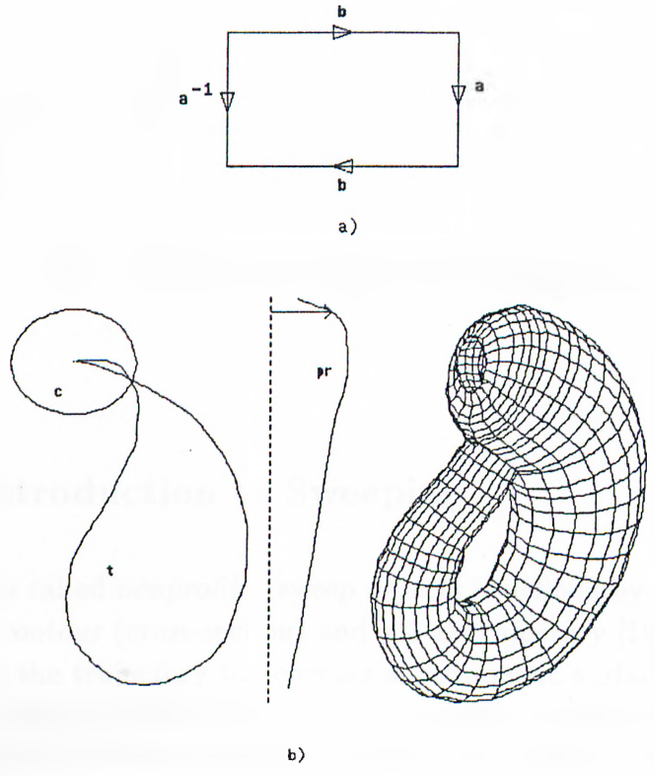


Figure 6 A Klein bottle is defined by a rectangle with $aba^{-1}b$ and by a contour c, a trajectory t, and a profile curve pr in $\mathsf{T}\flat$.

(Figure 6a), we obtain a tube for which the two end circles are denoted by b. But to identify the two end circles, it is necessary to match the arrows on them. Deform the tube so that one of the ends is somewhat smaller than the other and penetrate this smaller end through the wall of the tube. Then bend the larger end a little toward the interior and the smaller end a little toward the exterior, and attach them according to the required mode of identification. We are left with what is commonly known as a *Klein bottle* [22, 37, 53].

Figure 6b shows the curve representation of a Klein bottle[¶] in $\mathsf{T}\flat$. The contour curve c is scaled by the help of the profile curve pr while it is swept along the trajectory curve t.

[¶]More complicated topological objects, e.g., an eight knot [25], can be obtained by using contour, trajectory, profile, depth-modulation, and twist curves, ruled curves, skinning, and blending operations in $\mathsf{T}\flat$ (cf. Chapter 5).

3 The Sweep Paradigm

3.1 An Introduction to Sweeping

A class of solids called *nonprofiled sweep objects* is defined by two parametric curves: a 2-D *contour* (cross-section) and a 3-D *trajectory* [18]. The contour is moved along the trajectory to generate a parametric surface. A classification of sweep objects is given by Choi and Lee [20]. Another classification is given by Bronsvoort, Nieuwenhuizen, and Post [18]. Here, we prefer the latter because it is more convenient for our purposes.

Sweep objects can be defined in increasing complexity as follows:

- Translational sweep: The contour is arbitrary and the trajectory is a straight line.
- Rotational sweep: The contour is arbitrary and the trajectory is a circle.
- Circle sweep: The contour is circular and the trajectory is arbitrary.
- General sweep (generalized cylinder): Both the contour and the trajectory are arbitrary. This can be classified as:
  - Nonprofiled general sweep,
  - Profiled general sweep,
  - Depth-modulated general sweep, and
  - Twisted-profiled general sweep.

Here, only general sweep will be considered.

3.1.1 Nonprofiled general sweep

Nonprofiled general sweep is defined by an arbitrary closed 2-D contour and an arbitrary 3-D trajectory [18, 50]. Here, the 2-D contour $\mathbf{c}$ can be defined in parametric form:

$$\begin{aligned} &\mathbf{c}(v) = (\mathbf{c}_x(v),\ \mathbf{c}_y(v)) \qquad v_1 \le v \le v_f \\ &\mathbf{c}(v_1) = \mathbf{c}(v_f) \end{aligned}$$

As a parameter, v varies from v_1 to v_f. The parametric functions $\mathbf{c}_x$ and $\mathbf{c}_y$ trace out the contour. The 3-D trajectory $\mathbf{t}$ can be defined in the parametric form:

$$\mathbf{t}(u) = (\mathbf{t}_x(u),\ \mathbf{t}_y(u),\ \mathbf{t}_z(u)) \qquad u_1 \le u \le u_l$$

As a parameter, u varies from u_1 to u_l. The parametric functions $\mathbf{t}_x$, $\mathbf{t}_y$, and $\mathbf{t}_z$ trace out the trajectory. For nonprofiled general sweep, the contour $\mathbf{c}$ moves along the trajectory $\mathbf{t}$. ($\mathbf{c}$ remains constant along $\mathbf{t}$.)

The spatial relation between the contour and the trajectory must be defined at every point of the trajectory. Figure 7 illustrates a nonprofiled general sweep object. The contour plane is perpendicular to $\mathbf{e}_1$, the unit vector tangent to the trajectory. As $\mathbf{e}_3$, a fixed unit normal to the plane of the trajectory is chosen. Note that $\mathbf{e}_2$ is the vector product of $\mathbf{e}_3$ and $\mathbf{e}_1$. A profiled general sweep object is given in Figure 8.

Consequently, it is possible to present a nonprofiled general sweep as a vector function $\mathbf{T}_{np}$ of two parameters u and v:

$$\begin{aligned} &\mathbf{T}_{np}(u,v) = (\mathbf{Sweep}(\mathbf{t}(u),\ \mathbf{c}(v)) \\ &u_1 \le u \le u_l \qquad v_1 \le v \le v_f \end{aligned}$$

For the sweep operation, it is necessary to use translation and 3-D rotation. The direction of the tangent vectors for the first and the last points of the trajectory must be selected as directions of the first and the last segments to rotate the contour. For other points, the direction of the vectors as specified by the previous and the next points must be selected [5].

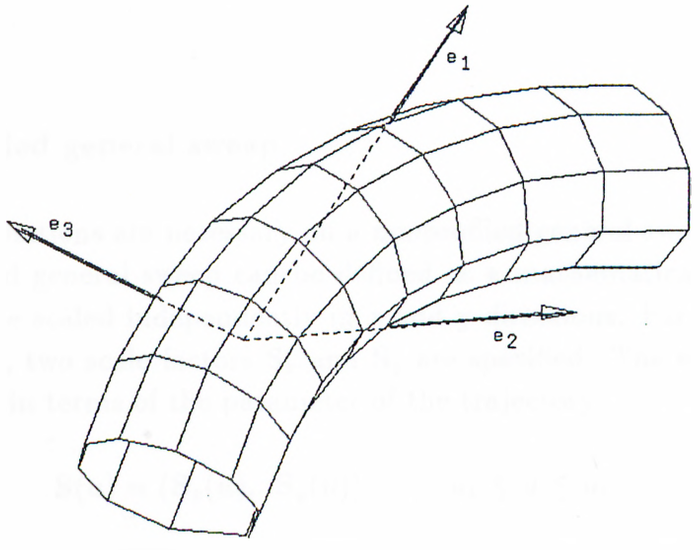


Figure 7 A nonprofiled general sweep object.

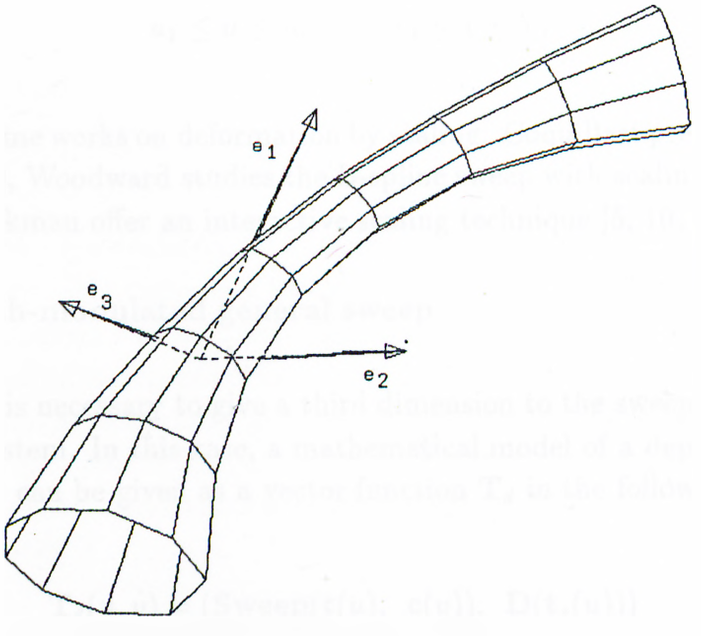


Figure 8 A profiled general sweep object.

Other mathematical models of sweep objects are given by Post and Klok [50], Martin and Stepson [44], and Wang [57].

3.1.2 Profiled general sweep

If some deformations are necessary on a nonprofiled general sweep by scaling, then a profiled general sweep can be defined as a mathematical model. The contour can be scaled independently in x and y directions. For each point of the trajectory, two scale factors $\mathbf{S}_x$ and $\mathbf{S}_y$ are specified. The scale functions are expressed in terms of the parameter of the trajectory:

$$\mathbf{S}(u) = (\mathbf{S}_x(u),\ \mathbf{S}_y(u)) \qquad u_1 \leq u \leq u_l$$

If the scale factor is substituted in $\mathbf{T}_{np}$ (cf. Section 3.1.1), it will have an effect only on the contour functions and the resultant profiled general sweep will be defined as a vector function $\mathbf{T}_p$:

$$\mathbf{T}_p(u,v) = (\mathbf{Sweep}(\mathbf{t}(u),\ \mathbf{c}(v)),\ \mathbf{S}(u))$$
$$u_1 \leq u \leq u_l \qquad v_1 \leq v \leq v_f$$

We cite some works on deformation by scaling. Coquillart presents an offset technique [21], Woodward studies the B-spline sweep with scaling [58, 59], and Arslan and Akman offer an interactive scaling technique [5, 10, 7].

3.1.3 Depth-modulated general sweep

Sometimes it is necessary to give a third dimension to the sweep objects in an interactive system. In this case, a mathematical model of a depth-modulated general sweep can be given as a vector function $\mathbf{T}_d$ in the following form:

$$\mathbf{T}_d(u,v) = (\mathbf{Sweep}(\mathbf{t}(u),\ \mathbf{c}(v)),\ \mathbf{D}(\mathbf{t}_z(u)))$$
$$u_1 \leq u \leq u_l \qquad v_1 \leq v \leq v_f$$

Section 4.2.2 gives more information about depth-modulated general sweep objects.

3.1.4 Twisted-profiled general sweep

If the contour curve is deformed by twisting and scaling while sweeping along the trajectory curve, the produced surface is called a twisted-profiled general sweep object. The contour curve can be twisted independently in x and y directions for each point of the trajectory curve. For this purpose, two twist factors $\mathbf{T}_{wx}$ and $\mathbf{T}_{wy}$ can be specified. The twist functions are expressed in terms of the parameter of the trajectory:

$$\mathbf{T}_w(u) = (\mathbf{T}_{wx}(u),\ \mathbf{T}_{wy}(u))$$
$$u_1 \leq u \leq u_l$$

If the twist factor is substituted in $\mathbf{T}_p$ defined in the previous subsection, it will have an effect only on the contour functions and the resultant twisted-profiled general sweep will be defined as a vector function $\mathbf{T}_{wp}$:

$$\mathbf{T}_{wp}(u,v) = (\mathbf{Sweep}(\mathbf{t}(u),\ \mathbf{c}(v)),\ \mathbf{T}_w(u),\ \mathbf{S}(u)))$$
$$u_1 \leq u \leq u_l \qquad v_1 \leq v \leq v_f$$

We now cite some works on deformation which influenced our work. Coquillart presents an offset technique for control points of rational B-spline curves to produce (profiled) sweep objects [21]. Choi and Lee use simple transformations and blending to represent more complex sweep objects [20]. Woodward presents a skinning technique to produce 3-D shapes that resemble blended sweep objects [58, 59]. Finally, Arslan and Akman present new interactive techniques to produce and deform sweep objects [5, 6, 7, 8, 9, 10, 11].

3.2 A Discrete Model for Sweeping

A sweep surface $Sw(C,T)$ is produced by moving a given contour curve C along a given trajectory curve T. The plane of C must be perpendicular to T at any point of T. C may be any planar closed or open curve and T may be any 3-D closed or open curve. If C is a closed curve and T is a 3-D curve segment, then the produced sweep object is called a *generalized cylinder* [21].

If C is deformed by scaling with a scale factor as it is swept along T, then the produced sweep object is called a *profiled general sweep* object. Here, we accept as the scale factor the distance of a given curve P to any selected axis. So, a profiled general sweep surface can be defined as $Sw_p(C,T,P)$. Here, P represents a profile curve which is the same number of points with T.

If C is deformed by twisting and scaling while sweeping along T, the produced object is called a *twisted-profiled general sweep object.* We define the twist factor by a curve Tw. This is a planar curve which has the same number of points with T and is mapped between the minimum and the maximum values of the twisting angles only at one dimension. Then, a twisted-profiled general sweep surface can be defined by four curves, viz. $Sw_{tp}(C,T,P,Tw)$.

In the following subsections, we present a discrete mathematical definition of twisted-profiled general sweeping. In this definition, we use the conventional matrix notation[‖] to represent the transformations and obtain a general sweep matrix as a result. In our representation, the matrix twists, scales, and sweeps a contour curve C along a trajectory curve T to handle a grid of general sweep surface points. We will formulate the discrete mathematical definitions of curves (C, T, P, Tw) as vector functions.

C is a planar curve or polygon with n points and each point $(x_i, y_i, 0)$ of C can be represented as C_i, $i = 1, 2, \ldots, n$. T is any 3-D curve or polygon with m points and (without loss of generality) the first point of T is in the origin. Each point (x_j, y_j, z_j) of T can be represented as T_j, $j = 1, 2, \ldots, m$. P can be given as a 1-D position vector with m component vectors that include scaling factors. But here, we take P as a planar curve and later calculate the scaling

[‖]Some matrix representations for sweeping are given by Rogers and Adams [54]. But, these are restricted and work only for given planar curves and situations that depend on curve equations.

factors by the help of the curve. This may be useful for interactive systems. Each point $(x_j, y_j, 0)$ of P can be represented as P_j, $j = 1, 2, \ldots, m$. Tw can be also given as a 1-D position vector with m vectors that include twisting factors. We choose to specify Tw as a planar curve and later calculate the twisting factors by the help of the curve. Each point $(x_j, y_j, 0)$ of Tw can be represented as Tw_j, $j = 1, 2, \ldots, m$. In the sequel, we give a procedure to handle a twisted-profiled general sweep object. In this procedure, C is a curve centered at the origin and the first point of T is also at the origin. In other cases, they must be translated to origin and then the generated curves must be translated back to their actual positions at the end of the procedure. First, C is rotated around the z-axis by Tw for twisting. Second, C is scaled by P. Third, C is rotated around the x, y, and z-axes by angles found for each point of T. Finally, each twisted, scaled, and rotated C is translated to every point of T.

3.2.1 Twisting

We have taken the twisting factor Tw as a planar curve. But, Tw represents only the twisting angles for each point of the trajectory curve T. That is, it can be represented as a 1-D vector function linearly interpolated at only one dimension of Tw. Tw can be drawn interactively as a planar curve in an interactive system and then is interpolated at one dimension to handle twisting angles (cf. Section 4.2.4 for details). The interpolated 1-D vector function θ includes rotating angles for each point of T and can be shown as follows θ_j, $j = 1, 2, \ldots, m$. The following matrix rotates a contour curve C that is centered at the origin (for twisting as a function of θ):

$$\left[R_{tw} \right] = \begin{bmatrix} c\theta_j & s\theta_j & 0 & 0 \\ -s\theta_j & c\theta_j & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} c\theta_j \stackrel{Df}{=} \cos\theta_j \\ s\theta_j \stackrel{Df}{=} \sin\theta_j \end{matrix}$$

3.2.2 Scaling

We have taken a profile curve P (a planar curve) as the scaling factor. P also represents only one magnitude for each point of the trajectory curve T. Then,

it can also be written as a 1-D vector function which takes the distances of the vertices of P to the vertical (or horizontal) axis at only one dimension. For an interactive system, first, an axis is given for reference and P is drawn interactively. Then, distances of vertices of P at one dimension to the axis give the scaling factor for C. Scaling factor s is a 1-D vector function and includes scaling radii for points of T. It can be shown as s_j, $j = 1, 2, \ldots, m$. The following matrix scales a contour curve C that is centered at the origin, as a function of s:

$$\left[S_p \right] = \begin{bmatrix} s_j & 0 & 0 & 0 \\ 0 & s_j & 0 & 0 \\ 0 & 0 & s_j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3.2.3 Rotation

The twisted-profiled contour curve C must be rotated in 3-D with respect to the tangent vector at each point of trajectory curve T. That is, the plane of C must be made perpendicular to the tangent vector of T at each point. We calculate the discrete derivative for each point of T to handle the tangent vector. To this end, we add two dummy vertices T_0 and T_{m+1} to T. ($T_0 = T_1$ and $T_{m+1} = T_m$.) In the following, we use matrices R_x, R_y, and R_z to handle the real position of C and Tx, Ty, and Tz to represent the x, y, and z distances of the vertices of T.

Rotation around the x-axis:

$$\left[R_x \right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_j & s\alpha_j & 0 \\ 0 & -s\alpha_j & c\alpha_j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{aligned} c\alpha_j &\stackrel{Df}{=} \cos\alpha_j = \frac{Ty_{j-1} - Ty_{j+1}}{h_x} \\ s\alpha_j &\stackrel{Df}{=} \sin\alpha_j = \frac{Tz_{j+1} - Tz_{j-1}}{h_x} \end{aligned}$$

$$h_x = \sqrt{(Ty_{j-1} - Ty_{j+1})^2 + (Tz_{j-1} - Tz_{j+1})^2}$$

Rotation around the y-axis:

$$\left[R_y \right] = \begin{bmatrix} c\beta_j & 0 & -s\beta_j & 0 \\ 0 & 1 & 0 & 0 \\ s\beta_j & 0 & c\beta_j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{aligned} c\beta_j &\stackrel{Df}{=} \cos\beta_j = \frac{Tx_{j-1} - Tx_{j+1}}{h_y} \\ s\beta_j &\stackrel{Df}{=} \sin\beta_j = \frac{Tz_{j+1} - Tz_{j-1}}{h_y} \end{aligned}$$

$$h_y = \sqrt{(Tx_{j-1} - Tx_{j+1})^2 + (Tz_{j-1} - Tz_{j+1})^2}$$

Rotation around the z-axis:

$$\left[R_z \right] = \begin{bmatrix} c\gamma_j & s\gamma_j & 0 & 0 \\ -s\gamma_j & c\gamma_j & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{array}{l} c\gamma_j \stackrel{Df}{=} \cos\gamma_j = \frac{Tx_{j-1} - Tx_{j+1}}{h_z} \\ s\gamma_j \stackrel{Df}{=} \sin\gamma_j = \frac{Ty_{j+1} - Ty_{j-1}}{h_z} \end{array}$$

$$h_z = \sqrt{(Tx_{j-1} - Tx_{j+1})^2 + (Ty_{j-1} - Ty_{j+1})^2}$$

3.2.4 Translation

Finally, C must be translated to each point of T to finish the sweeping operation. We use a 3-D translation matrix as follows:

$$\left[T_{xyz} \right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & 1 \end{bmatrix} \quad \begin{array}{l} a \stackrel{Df}{=} Tx_j - Tx_1 \\ b \stackrel{Df}{=} Ty_j - Ty_1 \\ c \stackrel{Df}{=} Tz_j - Tz_1 \end{array}$$

3.2.5 Obtaining the matematical model

A general twisted-profiled sweeping matrix Sw_{tp} is obtained, if the above transformation matrices are multiplied in the following order:

$$\left[Sw_{tp} \right] = \left[R_{tw} \right] \left[S_p \right] \left[R_x \right] \left[R_y \right] \left[R_z \right] \left[T_{xyz} \right] .$$

After some manipulations, the following matrix is obtained:

$$[\ Sw_{tp}\ ] = \begin{bmatrix} s_j c\theta_j c\beta_j c\gamma_j & s_j c\theta_j c\beta_j s\gamma_j & -s_j c\theta_j s\beta_j & \\ +s_j s\theta_j s\alpha_j s\beta_j c\gamma_j & +s_j s\theta_j s\alpha_j s\beta_j s\gamma_j & +s_j s\theta_j s\alpha_j c\beta_j & 0 \\ -s_j s\theta_j c\alpha_j s\gamma_j & +s_j s\theta_j c\alpha_j c\gamma_j & & \\ -s_j s\theta_j c\beta_j c\gamma_j & -s_j s\theta_j c\beta_j s\gamma_j & s_j s\theta_j s\beta_j & \\ +s_j c\theta_j s\alpha_j s\beta_j c\gamma_j & +s_j c\theta_j s\alpha_j s\beta_j s\gamma_j & +s_j c\theta_j s\alpha_j c\beta_j & 0 \\ -s_j c\theta_j c\alpha_j s\gamma_j & +s_j c\theta_j c\alpha_j c\gamma_j & & \\ s_j c\alpha_j s\beta_j c\gamma_j & s_j c\alpha_j s\beta_j s\gamma_j & & \\ +s_j s\alpha_j s\gamma_j & -s_j s\alpha_j c\gamma_j & s_j c\alpha_j c\beta_j & 0 \\ a & b & c & 1 \end{bmatrix}$$

Consequently, a twisted-profiled sweep surface can be calculated in the following form. First, each point of C must be evaluated for matrix Sw_{tp}, and then, a new matrix Sw_{tp} must be modified for each incremented j.

$$\left[\ Sweep\ surface\ \right]_{i,j,\theta,s} = \left[\ C\ \right]_i \left[\ Sw_{tp}\ \right]_{j,\theta,s} \qquad 1 \leq i \leq n \qquad 1 \leq j \leq m$$

In the above formula, j varies slowly, i.e., we compute the formula for a particular j and every i.

For the planar case, if C is a 2-D curve or a polygon centered at the origin in the x-y plane and T is a curve or polygon starting at the origin in the x-z plane, then a planar twisted-profiled sweep surface can be computed with the help of following matrix:

$$\left[\ Sw_{ptp}\ \right] = \left[\ R_{tw}\ \right] \left[\ S_p\ \right] \left[\ R_y\ \right] \left[\ T_{xz}\ \right] .$$

After some manipulations, the following is obtained:

$$[\ Sw_{ptp}\ ] = \begin{bmatrix} s_j c\theta_j c\beta_j & s_j s\theta_j & -s_j c\theta_j s\beta_j & 0 \\ -s_j s\theta_j c\beta_j & s_j c\theta_j & s_j s\theta_j s\beta_j & 0 \\ s_j s\beta_j & 0 & s_j c\beta_j & 0 \\ a & 0 & b & 1 \end{bmatrix}$$

3.2.6 Sweeping with varying contour

A twisted-profiled sweep surface can be calculated in the following form. First, each point of C must be evaluated for matrix Sw_{tp}, and then, a new matrix

Sw_{tp} must be modified for each incremented j:

$$\left[\ Sweep\ surface\ \right]_{i,j,\theta,s} = \left[\ C\ \right]_i \left[\ Sw_{tp}\ \right]_{j,\theta,s} \qquad 1 \le i \le n \qquad 1 \le j \le m$$

In the above formula, j varies slowly, i.e., we compute the formula for a particular j and every i. The contour curve C is a constant curve along the trajectory curve T. But frequently C must be a different curve for some points of T to handle more complex sweep objects. Our approach to resolve this situation is as follows.

First, different contour curves are defined for some points of T. Second, these curves are linearly interpolated respectively to obtain new contour curves for each point of T. Third, the obtained 2-D discrete curves are used for the sweep operation. (That is, a different contour curve is used for each point of T to obtain sweep surfaces with varying contour.)

A twisted-profiled sweep surface with varying contour can be calculated in the following form:

$$\left[\begin{array}{c} Sweep\ surface \\ with\ varying\ contour \end{array}\right]_{i,k,j,\theta,s} = \left[\ C\ \right]_{i,k} \left[\ Sw_{tp}\ \right]_{j,\theta,s} \qquad \begin{array}{l} 1 \le i \le n \\ 1 \le j,k \le m \end{array}$$

In the above formula, a different contour curve is taken into account for each point of the trajectory curve.

4 Software Characteristics of Tb

4.1 User Interface of Tb

A good interactive modeling system must have an enjoyable and powerful user interface. In this section, the user interface system of Tb and some ideas on which the system is based will be presented.

4.1.1 User interface systems

The following short descriptions may be useful:

- *User interface management system*: A user interface management system (UIMS) is a tool set designed to encourage cooperation in the rapid development, tailoring, and management of the interaction in an application domain across varying devices, interaction techniques, and user interface styles [42].
- *User interface system*: The combination of window-based displays, menus, icons, and a mouse that is increasingly used on personal computers and workstations.
- *Interactors*: They are elements that are presented to the programmer by user interface systems and play an effective role to provide user computer interaction. In general, interactors include the following elements:
  - windows,
  - active or passive text areas,

- pulldown menus,
- buttons,
- mouse, light pen, and other pointing devices,
- scrollbars,
- etc.

Figure 9 shows some interactors of the Sunview** user interface system.

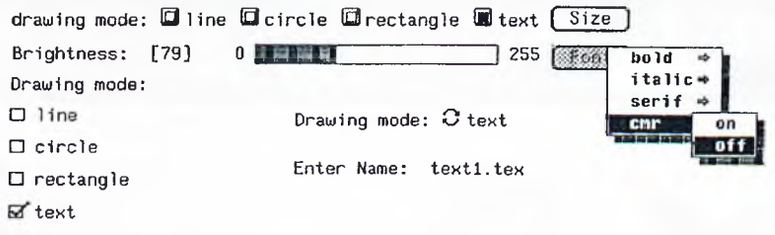


Figure 9 Some interactors of the Sunview user interface system.

- *Dialog*: A sample user interface, or part of a complex user interface, can be realized by a small, fixed collection of interactors. Such a collection is called a dialog [19].

4.1.2 User interface design of Tb

As we have mentioned in Section 4.1.1 a user interface management system presents a set of user interface components to programmers. In general, these components are called interactors (windows, menus, scrollbars, buttons, text areas, mouse, and so on). A programmer must only perform a suitable organization of these interactors. In other words, a programmer must build a user interface system by using events, states, or attributes of the supplied interactors.

A good user interface system must have the following properties [39]:

**Sunview is a registered trademark of Sun Microsystems, Incorporated.

- Ease of use and naturalness
- Speed and effectiveness
- Elegance and enjoyable features

Tb has a rather straightforward user interface. Figure 10 shows the general appearance of this interface.

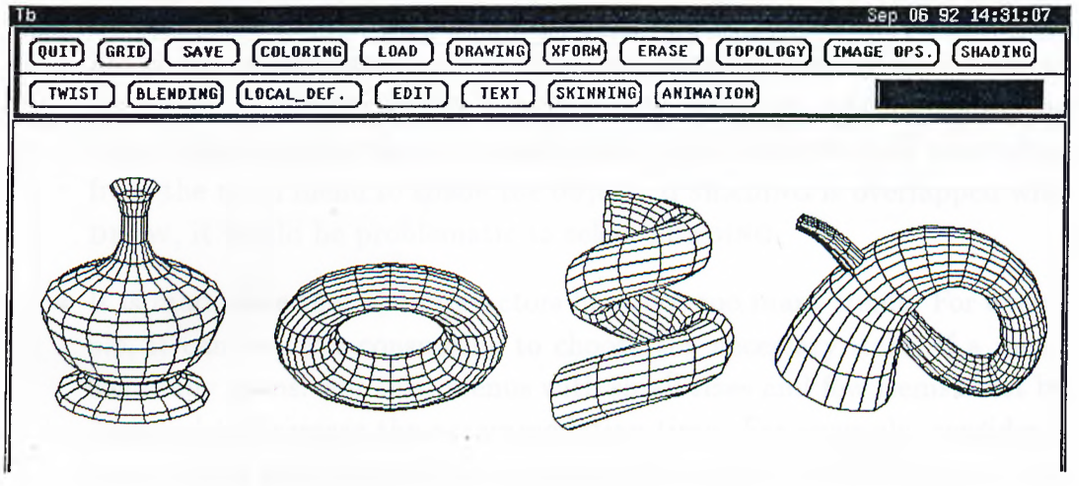


Figure 10 General appearance of the user interface system of Tb and some 3-D sweep objects produced by Tb.

Of course, the main goal of a user interface system must be to provide the above requirements. But Tb is a research tool and sometimes, speed is preferred over comprehensibility and orthogonality. Tb's user interface is based on the following guidelines:

- Interactors with major functionality are shown on the screen all the time. Some interactors with similar purposes were collected together. They were designed as buttons and placed in the top side of the main window. They are continuously shown on the screen so that the user immediately can activate any of them.

- Excluding text operations, only the mouse is used for interactions. Interaction time is important in our system; interaction must be provided in the shortest way. Using mouse was thus usually preferred over the keyboard.

- It is made sure that the interactors do not occupy too much space on the screen. The working area must be fairly wide and the user interface area must be fairly scant in a graphical interaction system.

- Interactors and their positions are good-looking and well-placed. Different colors, shapes, and fonts have been used for interactors to decrease the accommodation time of the user. The position of interactors must be in the adequate form. For example, consider the button DRAW to construct an object. Many submenus are shown to select the drawing type. After constructing the object, the button SHADING will be selected from the main menu to shade the object. If SHADING is overlapped with DRAW, it would be problematic to select SHADING.

- It is made sure that the interactors have not too many items. For example, it can be time consuming to choose the necessary items if a menu has many items. For this, menus with small sizes and few items must be designed to decrease the accommodation time. For example, consider a menu which has eight different drawing items and five different shading items. User has to select any of the 13 different items. If shading is not necessary at that moment, five items occupy space on the screen.

- To increase the speed of the system, sometimes interactors with a special purpose are defined. Figure 11 shows some rotational interactors in Tb.

- Some dynamic menus can be improved instead of using the keys of the keyboard. For example, loading is performed dynamically in Tb. The saved file names are shown immediately by color menus on the screen. User must select a file name to load it.

- There are two types of help menus used in Tb. In the first type, help menus are resident and tell the user what they will do. In the second, each interactor has its own help text which flashes or beeps to warn the user. Figure 12 gives some examples.

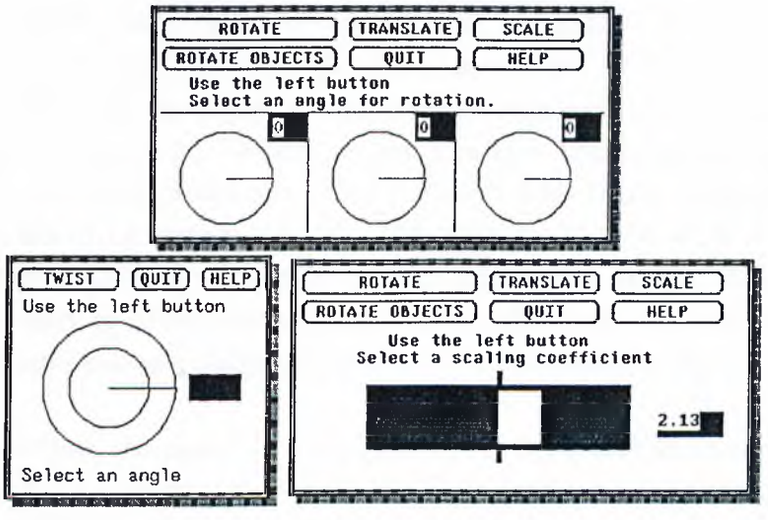


Figure 11 Some improved interactors (for rotation and scrolling).

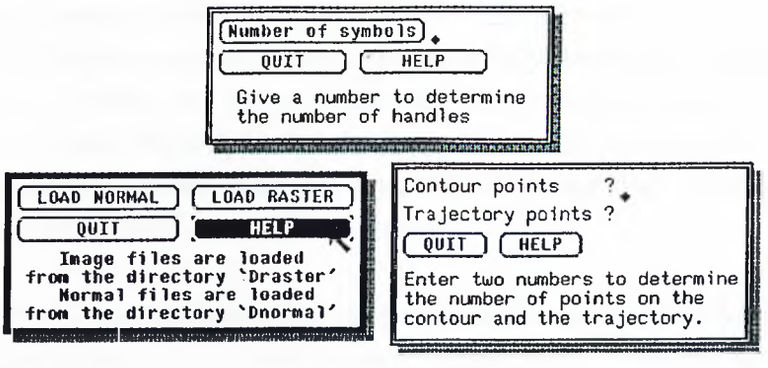


Figure 12 Three example help menus from Tb.

4.2 Available Functionalities

The implemented program Tb can be regarded as an interactive mathematical picture 'modeler', 'constructor', 'shader', and 'animator'. It is also useful for solid modeling [32, 33, 43, 52, 61]. The interaction of Tb with the user is straightforward. For example, a torus can be rendered by selecting only five control points to produce the contour and a radius. Our fundamental aim is to hide the details of the processing from the user. This is in the precise spirit of Pentland's sketching system [48, 49]. The main goal of our work is to provide a rudimentary graphical workbench to help topologists, artists, solid modelers, and geographers to illustrate their ideas more effectively. An auxiliary goal is to improve publication-quality production of mathematical figures.

In this section, the power of Tb is presented. A general appearance of main interactors of Tb is shown in Figure 10. Each interactor has many subinteractors. Therefore, each main interactor of Tb is given as a subsection and the respective submenus are presented in some detail.

4.2.1 Grid selection

We are interested in discrete sweeping and accordingly, every curve used in Tb is considered to be a sequence of discrete points.

A general sweep object is represented by two curves (contour and trajectory) in Tb. Contour and trajectory curves usually have many discrete points. The number of points on the contour and the trajectory curves determines the number of grids of the produced sweep object. So, the number of discrete points in auxiliary curves must be determined by the user. Menu GRID (cf. Figure 13) lets the user select this number.

The profiled, depth-modulated, and twisted sweep objects have a profile, a depth modulation, and a twist curve, respectively. The number of points on these curves is set automatically to the number of points of the trajectory curve. The help menu gives information about how the selection is done and how many points there are on the contour and the trajectory curves.

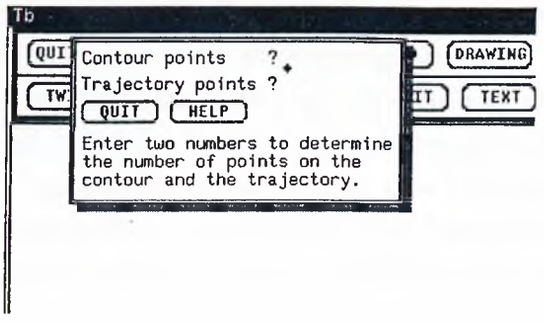


Figure 13 The number of points on the contour and the trajectory is selected by the user.

4.2.2 Drawing

In this subsection, we present some interactive methods to produce swept shapes. For each method, curves are specified and drawn interactively by the help of a mouse. They can be produced in two ways: free form and approximated form. In the former, a curve is produced by entering points of the curve by the mouse. There is no further operation to smooth the curve. In the latter, a curve is produced by entering each control point of the curve by the mouse. In this case, the curve is approximated with the use of Bézier or B-spline [17, 18, 19, 20, 21, 22] algorithms (and the produced curve is smooth).

The following sweep objects can be created by the help of drawing menu:

- Rotational sweep objects
- Nonprofiled sweep objects
- Profiled sweep objects
- Depth-modulated sweep objects
- Twisted-profiled sweep objects
- Assorted knots

In the following, We only give directions for creating a nonprofiled sweep object in detail by using the button DRAWING. Each of the other objects can be

created via similar selections.

Inputs to produce a nonprofiled sweep object are two curves. The output is a 3-D object obtained by moving or sweeping one curve along the other (cf. Figure 14). The method includes both translational and rotational sweep and provides the user with an opportunity to create complex 3-D objects. Curves that are sequences of points in space are stored in a 1-D array in Tb. They can also be determined in any other way such as a B-spline method or by inputting a sequence of 2-D points. One curve is considered to be a path on which the

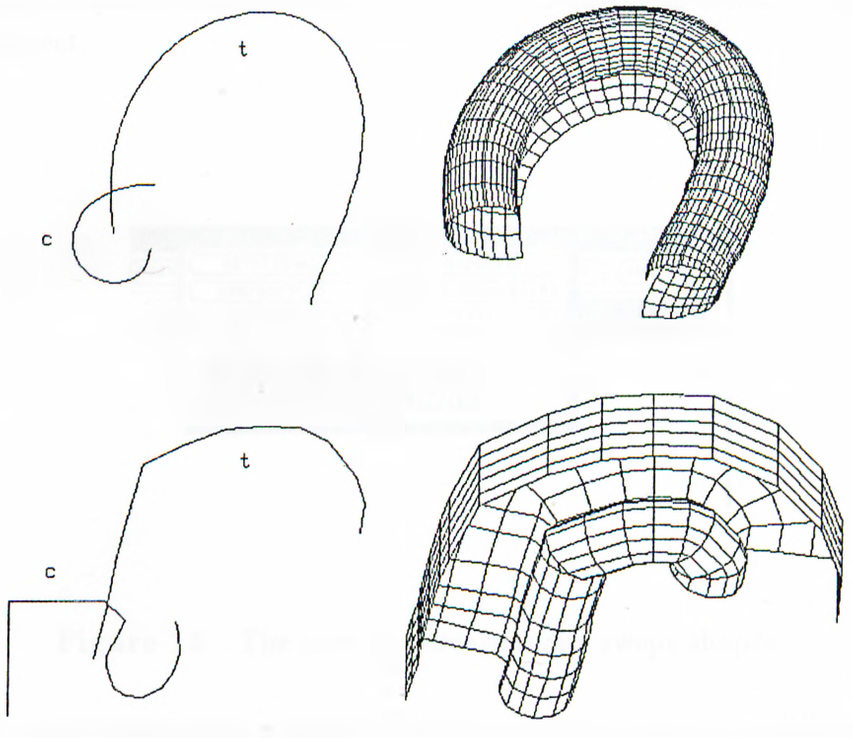


Figure 14 A contour is swept along a planar trajectory in order to generate a surface of desired shape. Both curves are defined arbitrarily.

other curve moves while generating a surface. The main idea of the method involves two transformations, namely a 3-D translation and a 3-D rotation. The first curve is replicated on the second curve by translating the contour to the points of the trajectory as many times as the number of points given for the trajectory. In other words, one copy of the contour curve is generated

for each point of the trajectory curve and is moved to these points. Next, the replicated curves are rotated by the position of the point.

The user must select the button DRAWING to produce a sweep object. After the selection of the button, the menu that is shown in Figure 15 is displayed. User must select the button NONPROFILED SWEEP to create a nonprofiled sweep object. Some submenus such as FREE_FREE, FREE_BEZIER, BEZIER_FREE, and BEZIER_BEZIER are shown on the screen. The button BEZIER_FREE inform that the contour curve will be drawn by using the Bézier algorithm and the trajectory curve will be drawn as a free form curve. After the selection of any menu, the contour and the trajectory curves of the nonprofiled sweep object is drawn. As a result, Tb uses curves to create a nonprofiled sweep object.

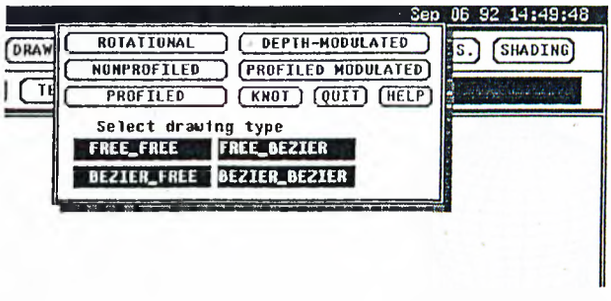


Figure 15 The user can create many swept shapes.

The second interactive method to produce swept shapes is well-known and commonly called *simple rotational sweep*. Let p be a point in 3-D. If p is rotated about an axis ℓ the resultant figure is a circle in 3-D. Similarly, if one takes a line segment s and rotates it about ℓ a cylindrical object is obtained. Now, take a circle c and rotate it about ℓ. The resultant figure is a torus (cf. Figure 16). Mouse-picked control points are used to define an arbitrary curve segment which in turn is rotated about ℓ. Many interesting surfaces of revolution can be computed in this way.

Although the above methods are restricted to 2-D curves, they can be extended to the 3-D case. For this, we give a third method called *nonplanar*

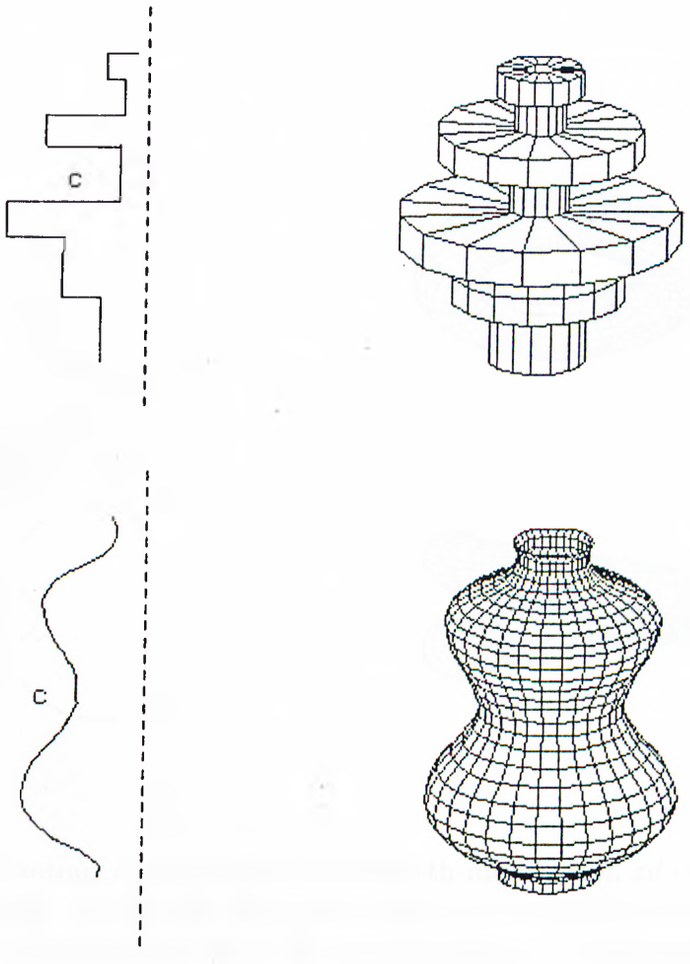


Figure 16 Both of the above objects are obtained by rotating a curve about an axis ℓ (shown in dotted lines). The first curve is a sequence of points, and the second one is a Bézier curve for which a set of control points is given.

general sweep. An important aspect of the method is to enter the 3-D points by the help of a mouse. This is provided by a third curve, appropriately called a *depth-modulation curve.* The curve gives a nonplanarity to the trajectory and now, the second curve is exactly a 3-D curve (cf. Figure 17). Some knots [22, 37], spirals, and other nonplanar complicated objects can be produced by the help of this method.

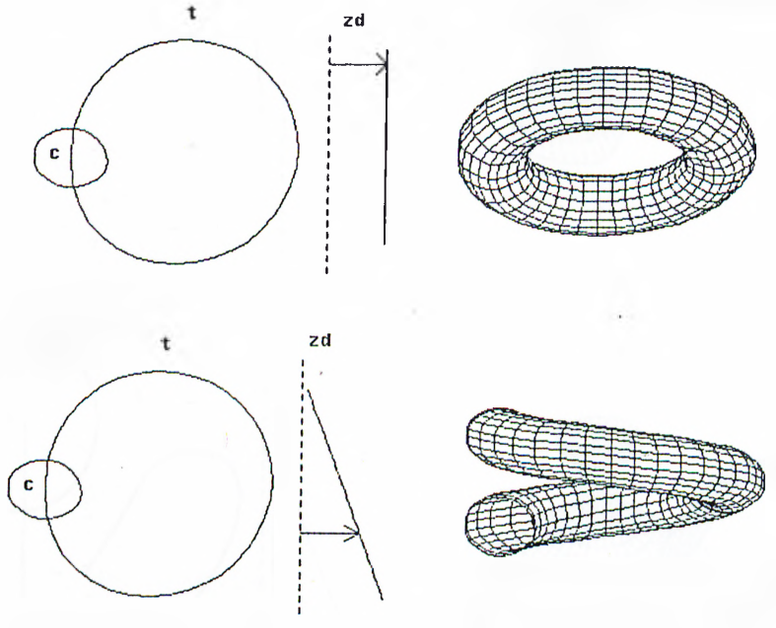


Figure 17 Contour c, trajectory t, and depth-modulation zd curves and the produced objects. In the first part, the distance of zd to the axis is constant and the resultant object is a torus. In the second part, this distance is varying and the resultant object is nonplanar.

The fourth method is *profiled general sweep.* Three curves are necessary for this method: a contour, a trajectory, and a profile curve. The contour curve is scaled to different sizes according to the profile curve while sweeping is performed (cf Figure 18).

The last method is a *twisted-profiled nonplanar general sweep.* Five curves are necessary for this method: a contour, a trajectory, a depth-modulation curve, a profile, and a twisting curve. The contour curve is twisted and scaled

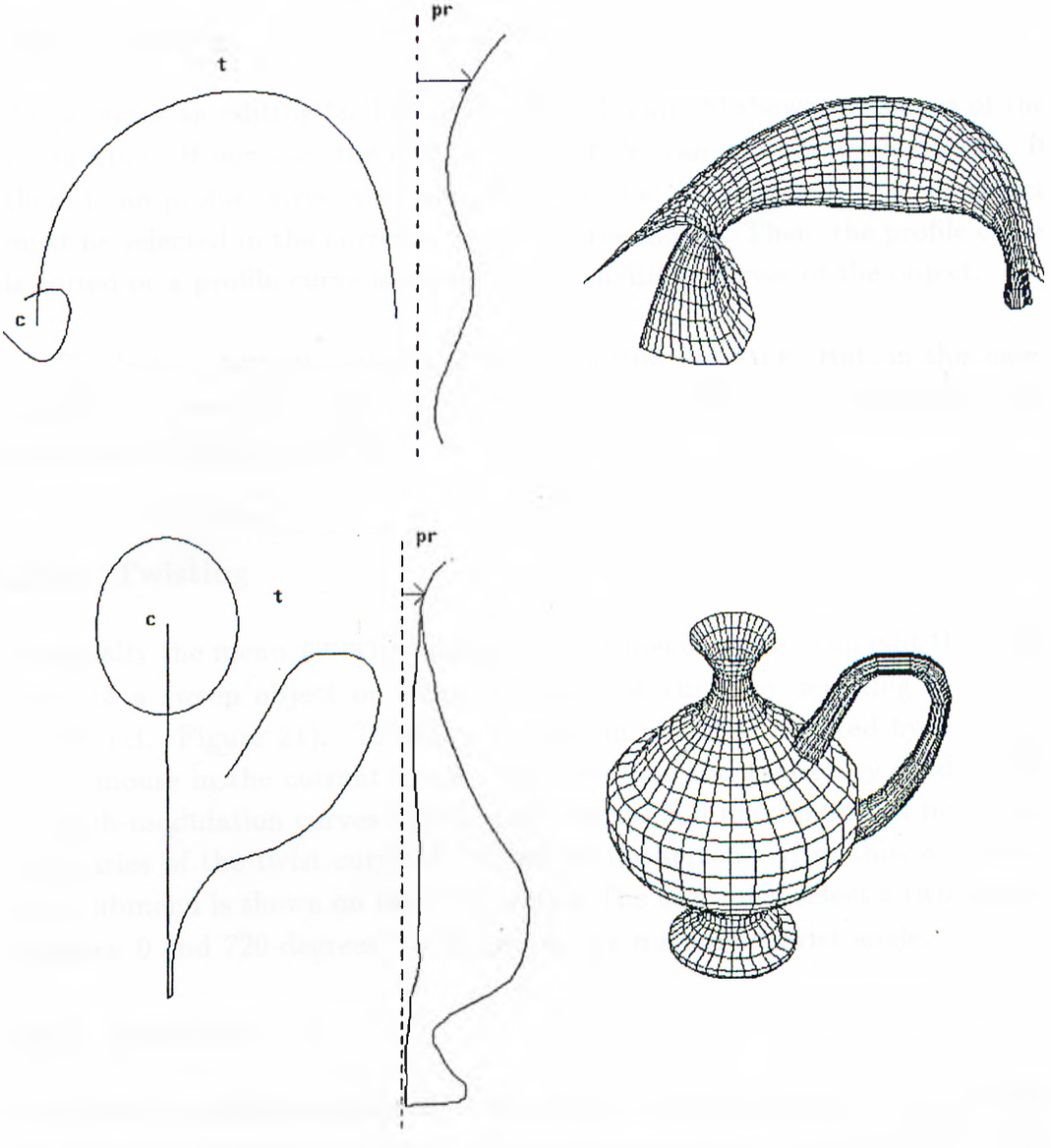


Figure 18 The contour curve is scaled to different sizes (via the curve *pr*) to obtain a profiled general sweep object.

to different angles and sizes according to the profile curve, while sweeping is performed (cf. Figure 19). Generation of the sweep surface finishes as soon as the points of the surface are computed. Some samples of produced sweep objects can be found in Chapter 5.

4.2.3 Editing

Tb provides an editing facility to the user. Figure 20 shows submenus of the menu EDIT. If one uses the button PROFILE, he can edit the profile curve. If there is no profile curve, the user can draw one. To this end, first an object must be selected in the current scene using the mouse. Then, the profile curve is edited or a profile curve is added to the auxiliary curves of the object.

The button DEPTH is the same as the button PROFILE. But, in this case, an object is selected in the current scene and the depth-modulation curve of the object is edited or created.

4.2.4 Twisting

Essentially the menu TWIST is also an editing menu. A user can edit the twist curve of a sweep object or create a new twist curve by selecting the twist button (cf. Figure 21). To this end, first an object is selected by the help of the mouse in the current scene. Then, contour and trajectory (and profile or depth-modulation curves, if necessary) curves and maximum and minimum boundaries of the twist curve are shown on the screen. After this, a circular twist submenu is shown on the twist menu. The user must select a twist angle (between 0 and 720 degrees) to determine the maximum twist angle.

4.2.5 Skinning

In the sweep methods mentioned in Section 3.2, the contour curve had constant shape along the trajectory curve. We give a new interactive method to obtain sweep objects with varying contour. At least three curves are necessary for this method. These are the contour curve, the trajectory curve and another contour curve. The number of contour curves is defined by user. That is, the user can draw many contour curves to determine the shape of the produced object. Two contour curves are linearly interpolated to obtain new contour curves for each point of the trajectory curve and then sweeping is performed by using these

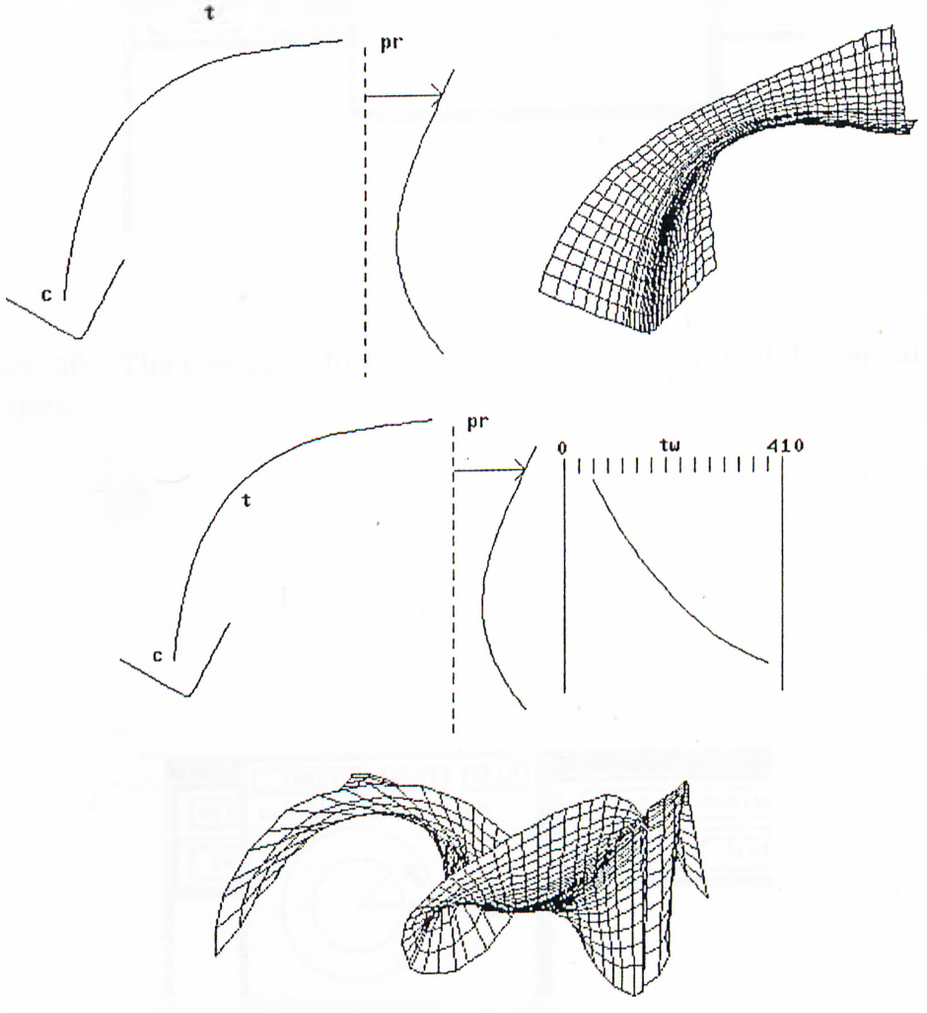


Figure 19 Contour c, trajectory t, profile pr, and twist tw curves and the produced objects. The first object is a profiled object. The second object is a twisted-profiled object.

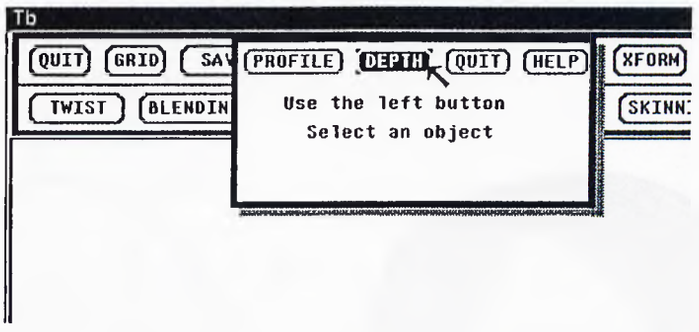


Figure 20 The user can edit the profile and the depth-modulation curves of an object.

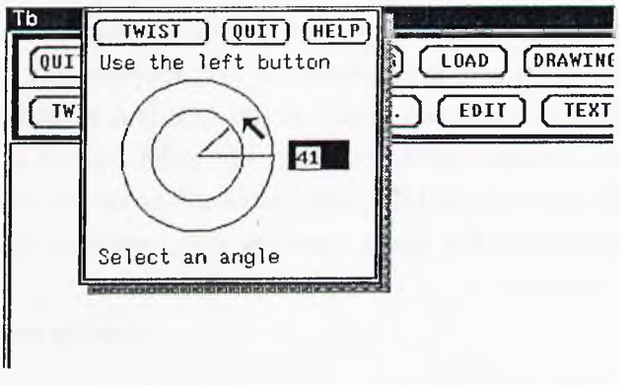


Figure 21 The user can twist a sweep object.

curves. If many contour curves are drawn, each curve is linearly interpolated to obtain new contour curves. The curves are swept along the trajectory curve and a new sweep object with different contour is produced (cf. Figure 22 for more information and Section 3.2.5 for a mathematical model of the object).

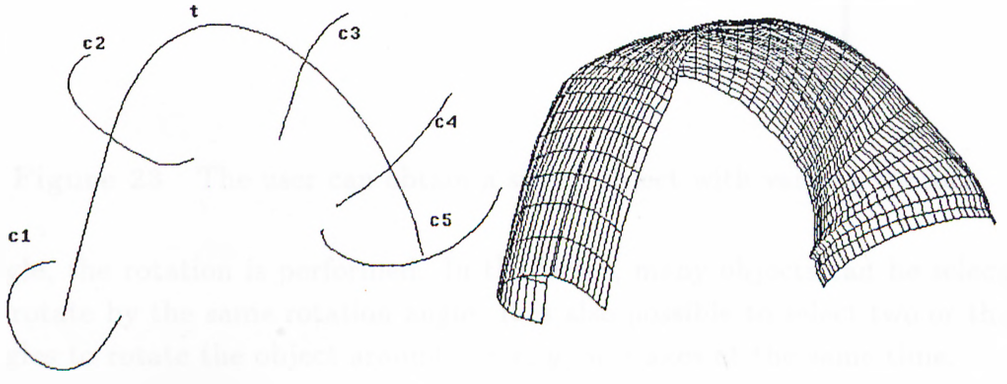


Figure 22 A trajectory curve and some contour curves with different shapes (c_1 to c_5) and the resultant sweep object with varying contour.

Users must select the button SKINNING to create an object with varying contours (cf. Figure 23). Then, the type of contour and trajectory curve must be selected from submenus displayed on the screen. If FREE_BEZIER is selected from menus, then the contour will be drawn as a free-form curve and the trajectory will be drawn as a Bézier curve. Each contour curve that is drawn will also be a free form curve. After the creation of the contour and the trajectory curves, the contour curves are linearly interpolated to obtain different contours and each produced contour curve is swept along the trajectory curve.

4.2.6 Transformations

Tb can perform 3-D rotation, scaling, and translation to give the required position or shape to the produced sweep object. The menu XFORM presents rotation, scaling, and translation options to the user (cf. Figure 24). Rotation can be performed in two ways. In the former, only one object is selected on the screen. After this, a rotation menu is displayed on the screen to select angles of rotation around x, y, or z-axis. After the selection of the rotation

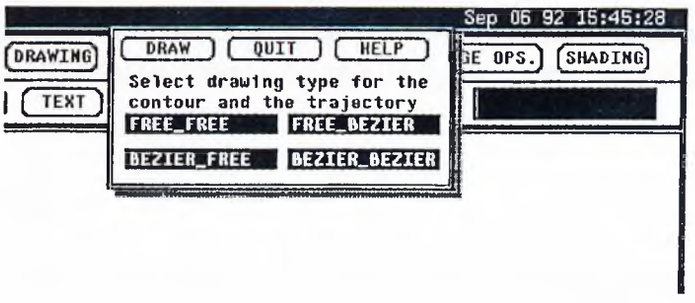


Figure 23 The user can obtain a sweep object with varying contour.

angle, the rotation is performed. In the latter, many objects can be selected to rotate by the same rotation angle. It is also possible to select two or three angles to rotate the object around the x, y, or z axes at the same time.

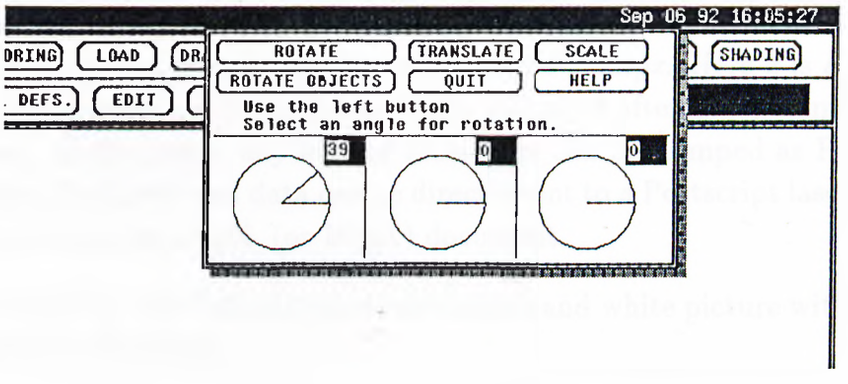


Figure 24 The user can perform 3-D transformations on a sweep object.

The button SCALE must be selected to scale an object. After selecting an object with the help of the mouse, a scrollbar is displayed on the screen. A number is selected on the scrollbar as the scaling coefficient. If the selected number is negative, then the selected object is made smaller. If the number is

positive, the object is enlarged.

For the translation of an object, the button TRANSLATE must be selected. The selected object is translated to the required place with the help of the mouse.

4.2.7 Shading

The generation of a sweep object finishes as soon as the points of the surface are computed. Then the resultant surface is shaded by constant or Gouraud shading [31, 47]. Constant shading is fast and adds enough realism to the obtained image. Hidden surfaces are eliminated using the z-buffer method [47]. Gouraud shading is slower than constant shading but provides a good enough effect toward realism for the generated surface.

A user must select the button SHADING (cf. Figure 25) to perform the following operations:

- The button AUX. CURVES draws auxiliary curves of a selected object on the screen. Thus, the user can see the auxiliary curves of an object.
- The button DUMP SCREEN gives two options to the user to dump the screen. In the former, the screen is dumped as hexadecimal numbers. The produced data is put into a file called Dump$x$. Here x is a number starting from 1 and is automatically increased after every dump operation. In the latter, any area of the screen can be dumped as Postscript code. The produced data can be directly sent to a Postscript laser printer or included in a TEX (or LATEX) document.
- The button WIREFRAME produces a black and white picture with hidden surfaces eliminated.
- The button CONSTANT shades a selected object by using a constant shading algorithm.
- There are two methods to shade a produced object by Gouraud shading. In the former, the button GOURAUD must be selected. Here, a special structure and sorting method is used to eliminate the hidden surfaces while shading is performed. This is a z-buffer method. After user selects an object on the screen by the help of the mouse, shading is performed. In the latter, the button SLIDE GOURAUD must be selected from the menu.

Maximum and minimum points of a selected object are computed in the four directions with respect to the center of the object in 2-D. A dynamic structure is constructed and each point of the subsurfaces of the object are sorted on the same coordinate of the structure. If we shade a selected object by using this method, we can create some windows showing any layer in the object by the help of the button SLIDE WINDOW.

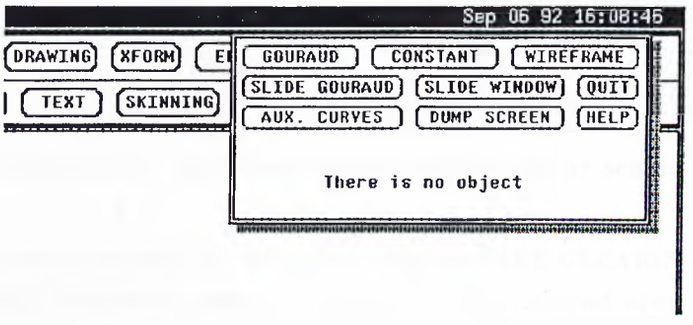


Figure 25 Some shading operations and screen dump can be performed.

4.2.8 Image operations

An image on the screen can be scaled and translated. Users must select the button IMAGE OPS. to perform image transformations (cf. Figure 26). If a user selects the button TRANSLATE, he must create a window that includes image which will be translated by using the left button. Then user can translate the selected image to any place on the screen by using the middle button.

If user selects the button SCALE, he must create a window on the image to be scaled. Then the user must select a scaling coefficient from the scaling interactor displayed on the screen.

4.2.9 Erasing and deleting

The user can erase a selected image or delete an object. Users must select the button ERASE from the main menu to perform these operations. After the selection, some menus are displayed on the screen (cf. Figure 27). A

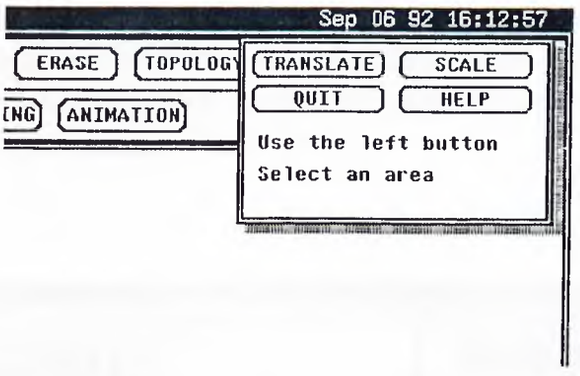


Figure 26 An image can be translated or scaled.

selected area can be erased by using the button FREE CLEARING. The button DELETE OBJECT deletes a selected object. If any colored area with different size is selected from the erasing window, a free form erasing can be done on the screen by using the mouse.

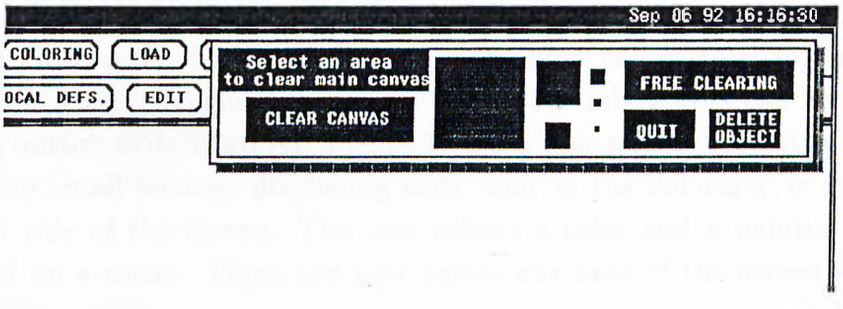


Figure 27 An image or an object can be deleted.

4.2.10 Text operations

The user can define any special character, assign the defined character to any key of keyboard, and display characters in any size and in any rotated position.

For this purpose, the user must select the button TEXT from main menus (cf. Figure 28). A special window is displayed on the screen if the user selects the button WRITE. User draws his own special character and the character can be assign to any key of keyboard. In addition, the selection of the size of characters and the selection of the rotation angle can be done by different interactors in the text menu.

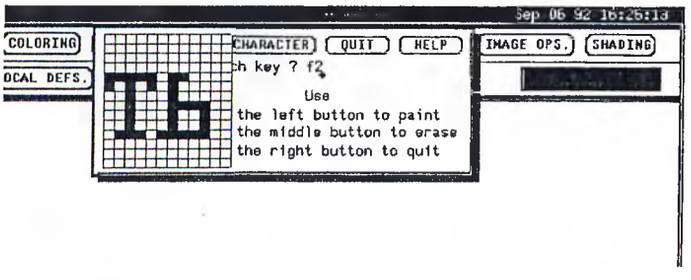


Figure 28 Special characters can be defined, assigned to any key of the keyboard, and displayed.

4.2.11 Coloring

Tb has an interactive free form painting, a dynamic colormap changing, and a background color changing interactor. The above interactors can be selected with the button COLORING (cf. Figure 29). If a user selects the button PAINTING, many small buttons displaying each color in the colormap is shown on the right side of the screen. The user selects a color and a painting area is presented on a menu. Then, the user paints any area of the screen by using the mouse.

User can change the background color of the screen by adjusting the values of red, green, and blue in the colormap by the help of the button BACKGROUND.

4.2.12 Save and load operations

The user can save or load an object or an image by the help of main menus SAVE and LOAD (cf. Figure 30). If the user wants to save an object, he or she

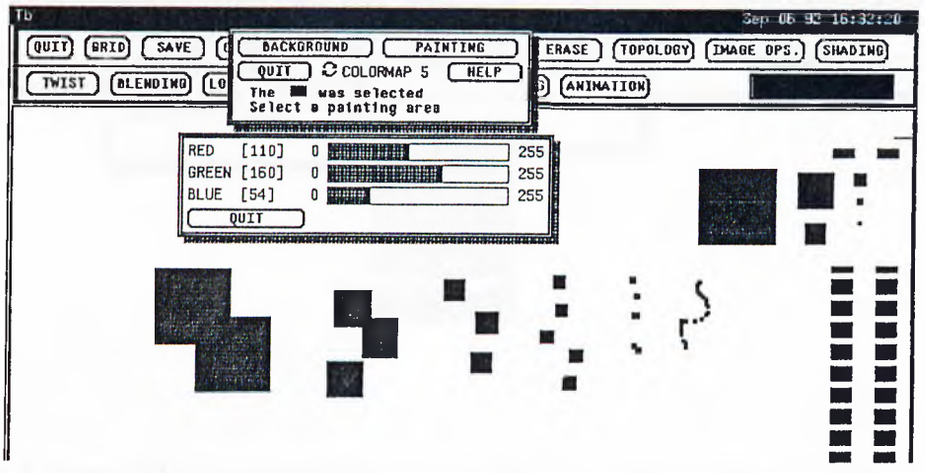


Figure 29 Tb has an interactive painting and colormap changing feature.

must write a file name into the text input panel. Then, the user must select an object using the the mouse. Tb saves 3-D coordinate values of each point of the object into the file. If the user wants to save an image, he must also give a raster file name and select an area on the screen that will be saved.

Each saved object or image can be loaded by using dynamic color menus. The user must select the button LOAD OBJECT or LOAD IMAGE from the menus of the button LOAD. A dynamic menu that includes the names of the saved files is displayed on the screen. The required file is selected from this menu.

4.2.13 Blending

If just one sweep operation is not sufficient to obtain a complicated object, two sweep objects can be blended to produce the required effect. Figure 31 shows the blending options in Tb. There are three interactors to perform different connections. Two different sweep objects can be blended by selecting one edge of each object if the user selects the first button. The user must draw a path between the selected edges. Only the control points of the curve must be given to obtain a smooth path. An approximated curve is obtained between the objects and sweeping is performed by using the curve and the edges of

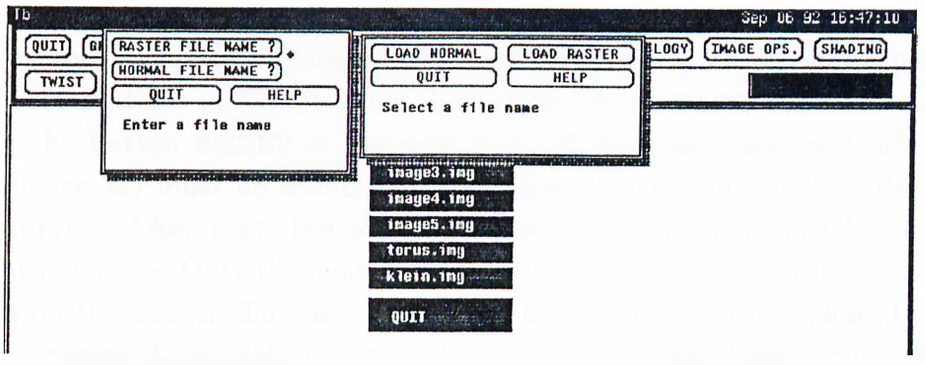


Figure 30 The user can save or load an object (or an image).

the objects. If the user selects the second button, the blending operation

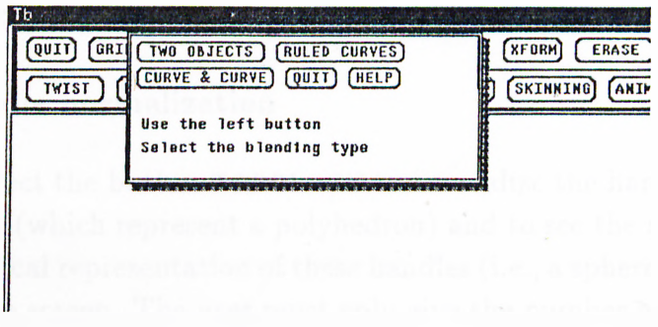


Figure 31 Some blending operations on sweep objects can be performed.

is performed between a curve selected on a sweep object and another curve drawn by the user. The user must also draw a path between the curves. A new object (with one edge on the selected object) is produced. Two curves with different shape are blended if the user selects the third button. The user must draw two different curves and a trajectory between these curves. Essentially, the performed operation is similar to the commonly used operation for ruled curves.

4.2.14 Local deformations

Sometimes it is necessary to perform local deformations on a sweep object to obtain more complicated objects. An easy method was developed to perform local deformations in Tb. For this, the button LOCAL DEFS. must be selected. Then, the button BEZIER or the button FREE from the displayed submenus (cf. Figure 32) must be selected to determine the type of auxiliary deformation curves. After that, the object that will be deformed must be selected using the mouse. Only the contour and the trajectory curves of the object are shown on the screen. The user must draw two auxiliary curves along the trajectory curves. Local deformation is performed using auxiliary curves. Figure 32 shows the contour c, the trajectory t, and deformation curves d_1 and d_2, and the locally deformed sweep object. All sweeping objects (profiled, depth-modulated, ...) can be locally deformed using this way.

We cite some works on deformation objects. Güdükbay and Özgüç use superquadrics and Bézier surfaces to model regular classes of objects and combine regular deformations and free-form deformation techniques to create objects with free-form surfaces [29]. Woodward gives methods to model and deform B-spline surfaces [58, 59].

4.2.15 Handle normalization

Users must select the button TOPOLOGY to normalize the handles in the produced symbols (which represent a polyhedron) and to see the normalized handles and graphical representation of these handles (i.e., a sphere with numerous handles) on the screen. The user must only give the number of topology symbols by using the topology menu (cf. Figure 33). Tb creates random symbols with the given number. Then, it computes the normal form of the symbols to obtain true handles. It prints the obtained handles and draws a sphere with these handles (cf. Figure 33).

Tb uses the classical algorithm given in Ringel [53] (also see Abelson and diSessa [1], Griffiths [27], and Stillwell [56]) to compute the normal form of a given polyhedron (cf. Section 2.1).

Similar to the way MACSYMA[††] provides users with high-level facilities such as matrix operations, integration, and solutions of differential equations, using

[††]MACSYMA is a product of the Symbolics, Inc.

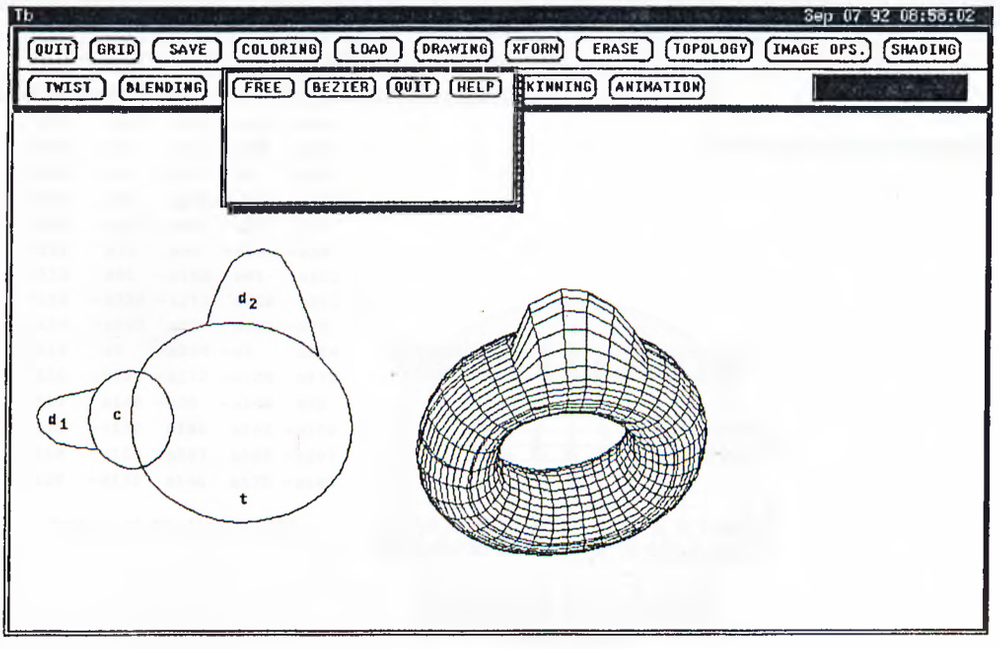


Figure 32 Local deformations on a sweep object. Tb locally deforms the object using d_1 and d_2 which are the first and the second local deformation curves, respectively.

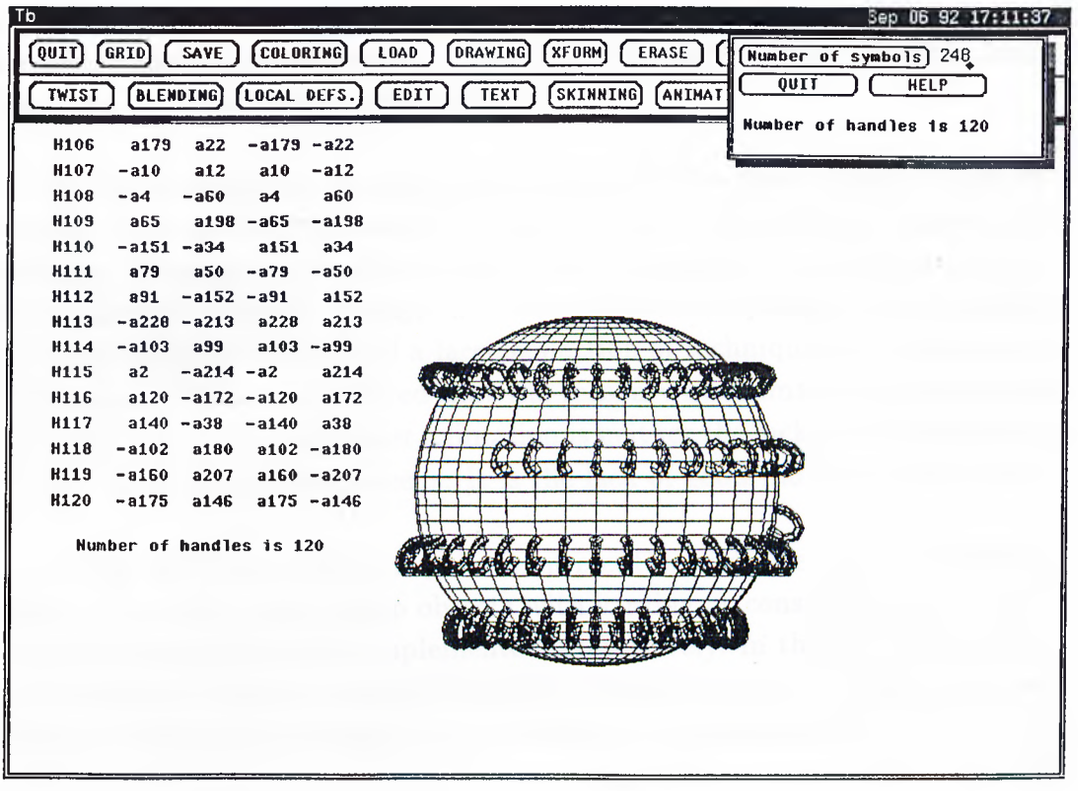


Figure 33 A sphere with 120 handles. Symbols show the normalized handles. For example, H_2 a_1 a_6 $-a_1$ $-a_6$ denotes the second handle obtained by the symbols a_1 and a_6. (The minus sign is used here as a synonym for $^{-1}$.)

Tb we aim to provide users with some high-level and simple concepts which are time-consuming and tedious to define. In our view the classification of 2-manifolds can be seen as such an operation.

4.2.16 Animation of sweep-based objects

The process of generating computer animation can be broken down into three phases [51]: *modeling*, *animation*, and *rendering*. In the modeling phase, the modeler creates a 3-D model of the scene. In the animation phase, the animator describes how the model change and a simulated camera move over time. In the rendering phase, the renderer takes the model and animation data and, for each frame in the sequence, computes a 2-D image of what the scene looks like from the viewpoint of the camera.

Of the three phases, rendering has attracted the most attention and research. New techniques needed to be developed to make the images look realistic. Rendering is computationally very expensive, so developing faster algorithms was critical. Today, there are much faster personal workstations, more efficient algorithms, and a large collection of techniques for making realistic images. Still, it is believed that the reason why animators are unable to produce high-quality computer animation is neither the lack of rendering techniques nor slow rendering speeds. It is the lack of good modeling techniques.

Tb has the three phases of animation mentioned above. In the modeling phase, Tb models many sweep objects with varying or constant contour. Animation of sweep objects is implemented interactively. In the rendering phase, it is necessary to render animated objects for each scene. In Tb, each produced object for animation is rendered as presented in the Section 4.7.

There are two ways for animating sweep objects in Tb. In the former, only one operation is performed for animation. In this case, the object to be animated is produced by using the modeler of Tb. Figure 34 shows the animation menu. In the latter, both modeling and animation are performed at the same time. If necessary, each produced object can be shaded.

The idea of animation of a sweep object is based on the linear interpolation of curves. Two animation curves are necessary to animate a sweep object. Each point of the curves is linearly interpolated and many new curves are produced. The first point of the first curve must be linearly interpolated to

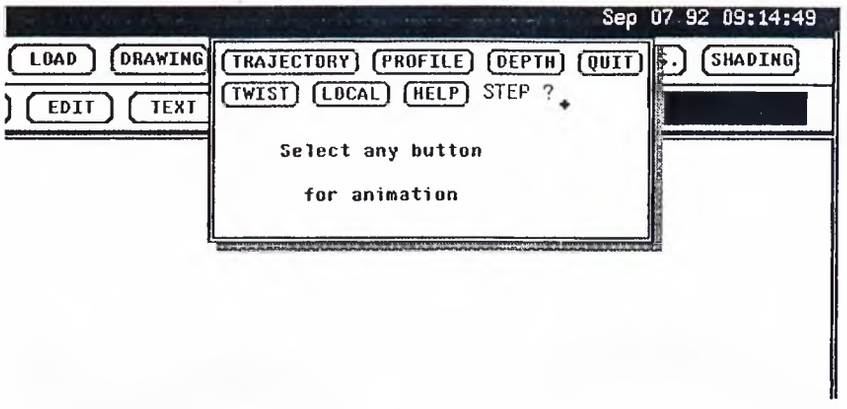


Figure 34 A sweep object can be animated.

the first point of the second curve, the second point of the first curve must be linearly interpolated to the second point of the second curve, and so on.

Figure 35 shows animation of a sweep object by the help of a trajectory curve. The curves t_1 and t_2 are first and the last forms of the trajectory curve, respectively. t_1 and t_2 are linearly interpolated and a sweep operation is performed for each produced curve. Of course, it is possible to animate a sweep object by the help of a contour curve. Two contour curves are interpolated and sweeping is performed for each produced contour curve. Figure 36 shows animation by the help of a profile curve (two profile curves are necessary). Curves are interpolated to produce new profile curves by an operation that is similar to the operation mentioned above. The contour curve is scaled by each produced profile curve while sweeping is performed. Figure 37 shows animation by the help of a depth-modulation curve. Two depth-modulation curves are interpolated and new depth-modulation curves are produced. The third dimension for each point of the trajectory curve is determined by each newly produced depth-modulation curve while sweeping is performed to animate a sweep object. Figures 38 and 39 show animation by the help of a twist curve. Twist curves are linearly interpolated and the contour curve is twisted by the help of each new twist curve. Here, we give simple animations with one sweep object. It is possible to use many sweep objects joined each other and to animate each sweep object.

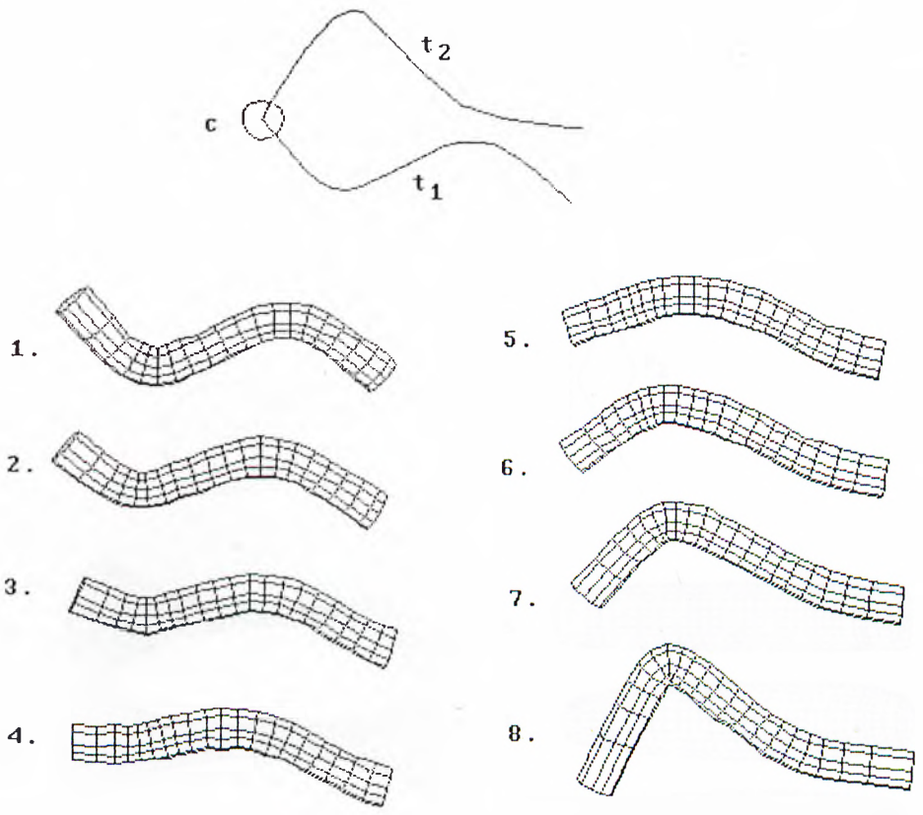


Figure 35 Animation by the help of the trajectory curve (c is the contour, t_1 and t_2 are the first and the last forms of the trajectory, respectively). Numbers show the animation sequence.

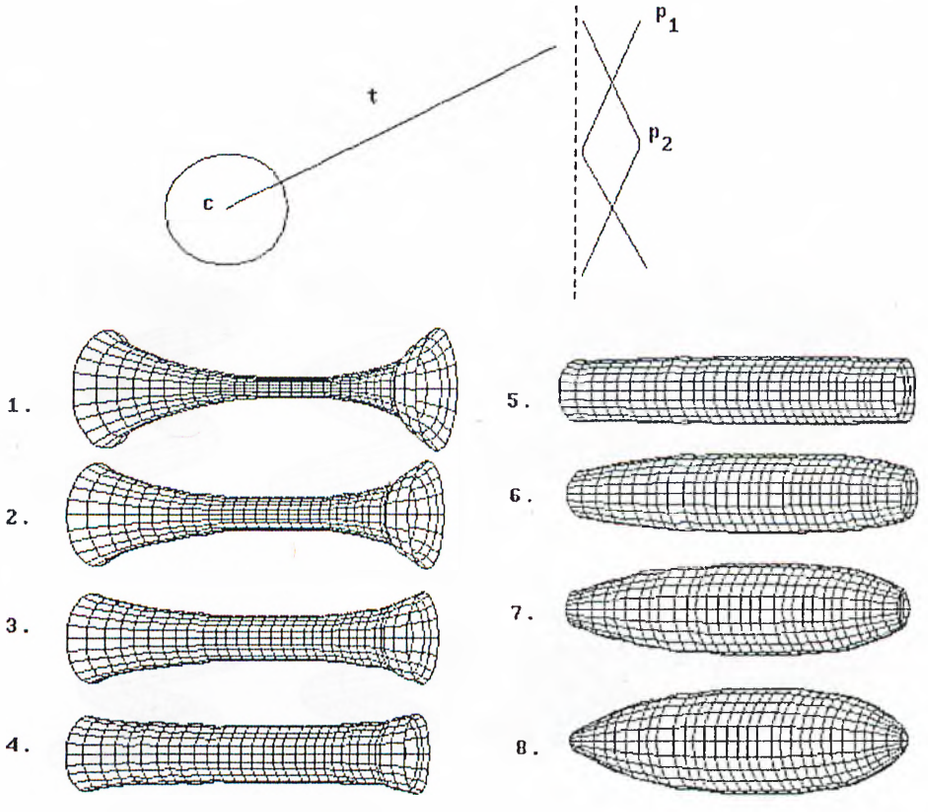


Figure 36 Animation by the help of the profile curve (c is the contour, t is the trajectory, and p_1 and p_2 are the first and the last forms of the profile, respectively). Numbers show the animation sequence.

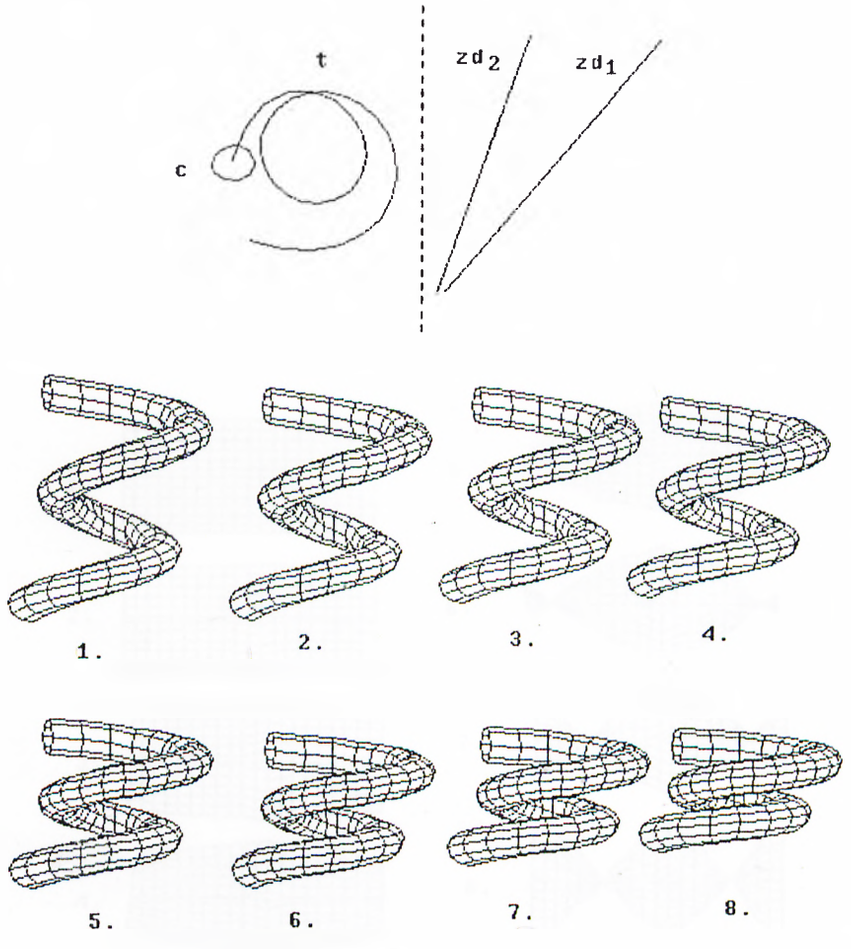


Figure 37 Animation by the help of the depth-modulation curve (c is the contour, t is the trajectory, and zd_1 and zd_2 are the first and the last forms of the depth-modulation curve, respectively). Numbers show the animation sequence.

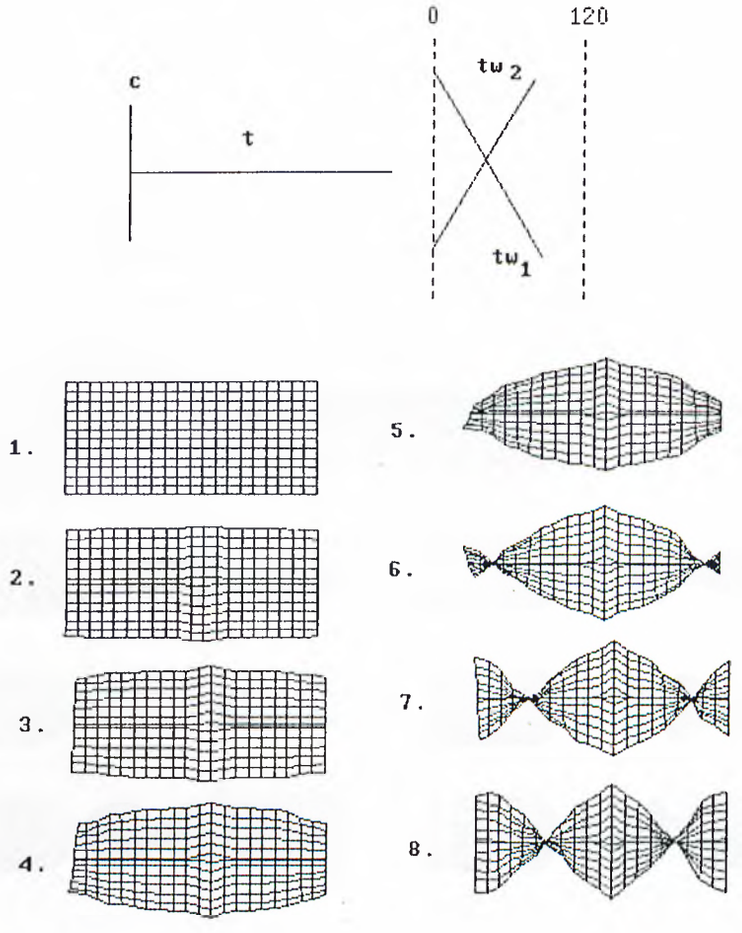


Figure 38 Animation by the help of twist curve (c is the contour, t is the trajectory, and tw_1 and tw_2 are the first and the last forms of the twist curve, respectively). Numbers show the animation sequence.

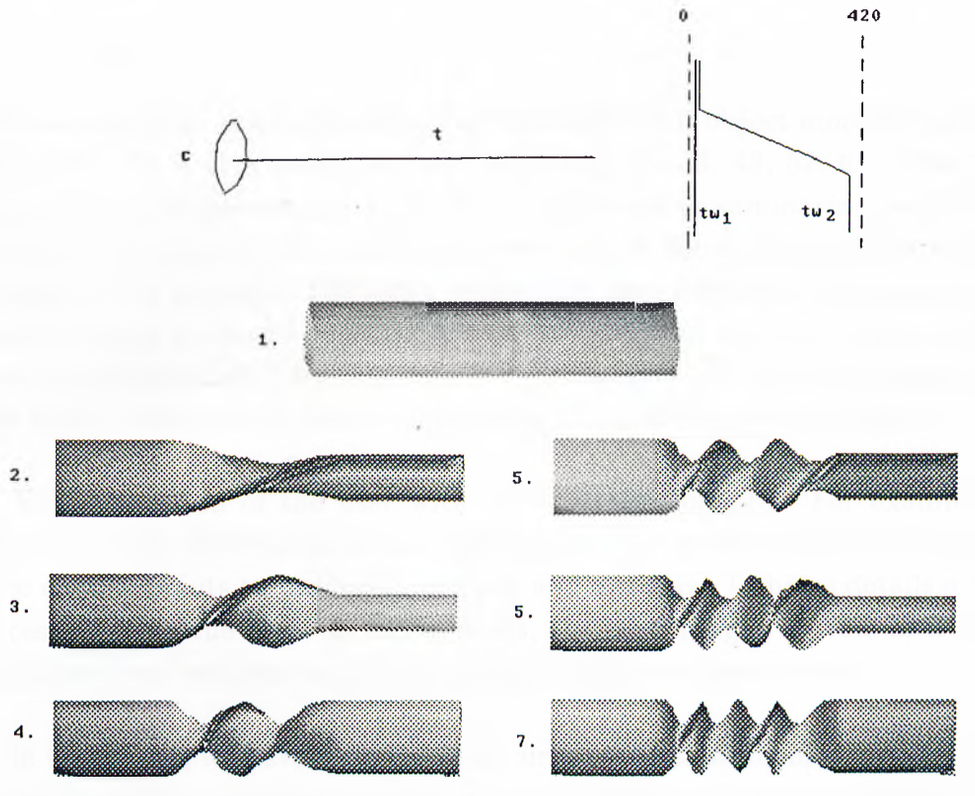


Figure 39 Local animation by the help of twist curve (c is the contour, t is the trajectory, and tw_1 and tw_2 are the first and the last forms of the twist curve, respectively). Numbers show the animation sequence.

5 Conclusion

We have designed and implemented an interactive 3-D object modeling system called Tb. Tb is also useful for solid modeling [32, 33, 43, 52, 61]. Previous approaches using sweeping to model 3-D objects can be summarized as follows. Surfaces of revolution (i.e. rotational sweep) can be found in any commercially available CAD package. The DUCT system [28] uses rotational sweep and nonprofiled sweep to model objects. The RAYMO system which is based on [55] includes rotational and nonprofiled sweep techniques. The PLAMO system [17] uses Frenet frame and rotation minimizing for modeling sweep objects.

The interaction of the user with Tb is straightforward. For example, a sphere (or any object similar to a sphere) can be rendered by selecting only three control points to produce a contour and a radius. Tb hides details of the processing from the user. In other words, the user designs objects as if he is doing free-form sculpturing by carrying out high-level operations.

In Chapter 2, we have compared the curve representation of a surface in Tb and its symbolic representation in topology. In Chapter 3, a new mathematical model was presented for the planar and the nonplanar twisted-profiled general sweep objects. In addition, the model was generalized for sweep objects with varying contours. Figure 40 shows some rotational sweep objects. Figures 41 and 42 show nonprofiled and profiled sweep objects, respectively. Figures 43 and 44 show some depth-modulated and twisted sweep objects, respectively. In Chapter 4, we have presented the user interface system and the available functionalities of Tb. Tb is an interactive system based on the Sunview windowing system. The following are some interesting aspects of Tb.

Global or local deformations can be performed interactively on a produced sweep object. For example, a small 'bump' or a tiny 'well' can be created on

a given surface, e.g., a torus. A locally or globally deformed object can be deformed repeatedly. Figure 45 shows some sweep objects with local deformations.

It is possible to produce sweep objects with a varying contour. Several contour curves with different shapes can be assigned interactively to the trajectory curve to produce the varying contour. Figure 46 shows some examples.

Transparent windows can be created on a sweep object to inspect the sublayers of it. We use a special data structure to open the windows on the object while shading is being performed. It is possible to create a rectangular (or a polygonal) window on the object. Figure 47 shows an object with transparent windows on it.

Animation of sweep objects is also an original study. Some auxiliary curves are used to animate the object.

Tb improves publication-quality production of mathematical figures as a color picture. In addition, it gives three options to the user to produce an image of an object for print out. First, the user can dump the screen for laser printers which accept data in hexadecimal format. Second, the user can dump any area of the black and white screen into Postscript code. Finally, any area of the color screen can be dumped into Postscript code. The dumped files can be included in a TEX (or LATEX) document or sent directly to a Postscript laser printer.

Sweep objects produced by Tb can be combined to produce a more complicated object. This can be done in two ways by Tb. In the former, the base sweep objects are produced and the objects are transformed to provide the required connections with each other. Figures 48, 49, and 50 show some connected objects produced in this way. In the latter, the sweep objects are produced and blended or connected with each other to produce a more complicated object. Some figures produced in this way are as follows.

Figure 51 (projections of the knot) was digitized from Francis' book (p. 150) and Figure 52 was produced by Tb. Figure 53 includes two pictures representing a knot. The first picture was digitized from Francis' book (p. 38) and the second was produced by Tb. Figure 54 (swapping handle cores) was digitized from Francis' book (p. 140) and Figure 55 was produced by Tb.

Figure 56 shows two pictures (eight knot). The first one was digitized from Francis' book (p. 37) and the second was produced by Tb.

There are some limitations of Tb. For example, Figure 57 was taken from [27, p. 87]. This figure cannot be easily produced by the help of Tb. It is not possible to build it as just one sweep object. Some parts of the statue can be easily produced by Tb; but blending would still be a problem. It is necessary to develop some higher level blending operations. If the user uses some auxiliary data files to enter contours and trajectories to obtain parts of the statue, then Figure 57 would be relatively easy to construct.

As a future work, some higher level blending operations may be developed to obtain more complicated objects; blending in Tb is naive and insufficient in its present form. Some other techniques (for example, B-spline surfaces with interactive operations, finite-element method, etc.) must be included to model complicated objects. Finally, a ray tracing option must be included in order to give a more realistic appearance to the produced objects. If a ray tracing method is included in Tb, it would be helpful to develop some parallel algorithms to reduce rendering time.

Figure 40 Some rotational sweep objects. The objects are represented with a contour curve and a radius in Tb. Each picture was computed by Tb in about 3 seconds after the initial design of the figures.

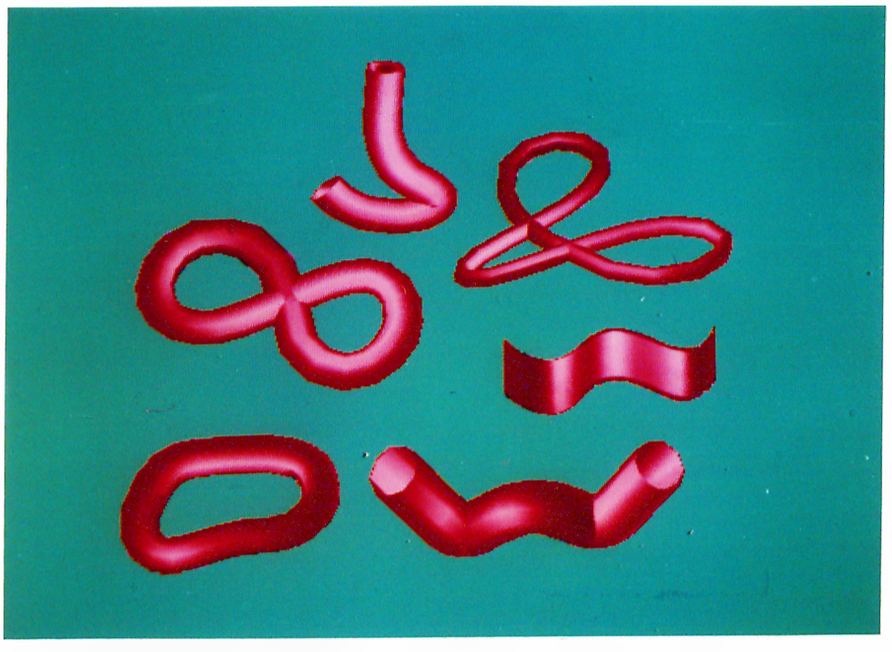


Figure 41 Some nonprofiled sweep objects. The objects are presented with a contour and a trajectory curve in Tb. Each picture was computed by Tb in about 4 seconds after the initial design of the figures.

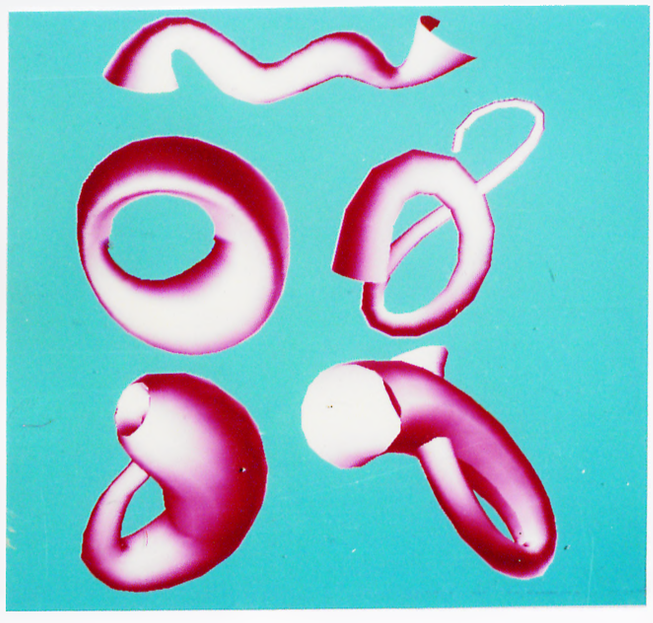


Figure 42 Some profiled sweep objects. The objects are represented with a contour, a trajectory, and profile curve in Tb. Each picture was computed by Tb in about 4 seconds after the initial design of the figures.

Figure 43 Some depth-modulated sweep objects. The objects are truly 3-dimensional and are represented with a contour, a trajectory, and a depth-modulation curve in Tb. It is the depth-modulation curve that gives the third dimension to the trajectory. Each picture was computed by Tb in about 4 seconds after the initial design of the figures.

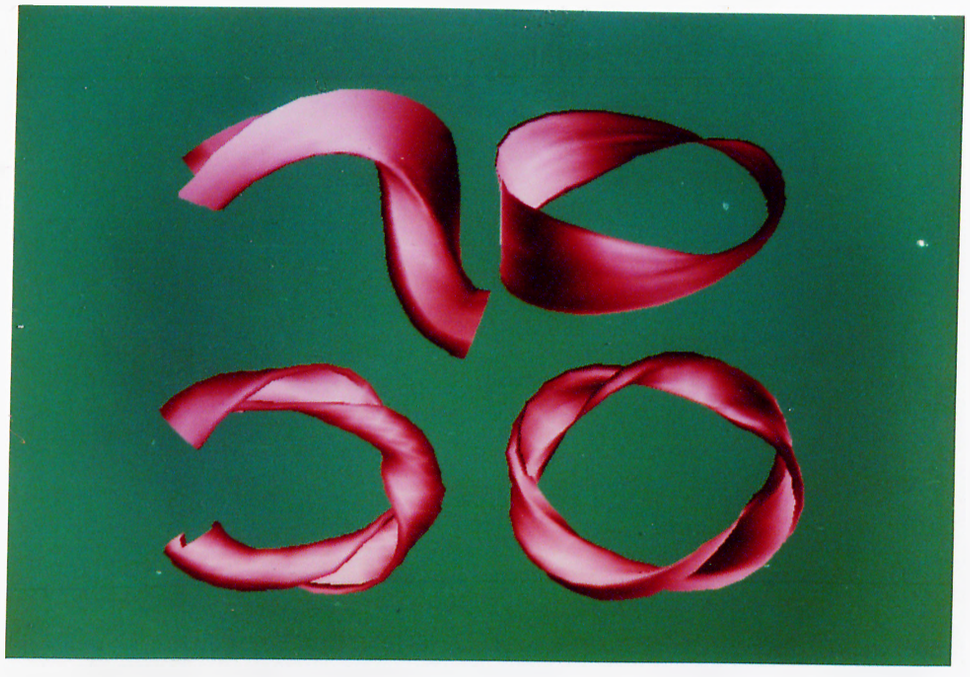


Figure 44 Some twisted sweep objects. The objects are represented with a contour, a trajectory, and a twist curve in Tb. Each picture was computed by Tb in about 5 seconds after the initial design of the figures.

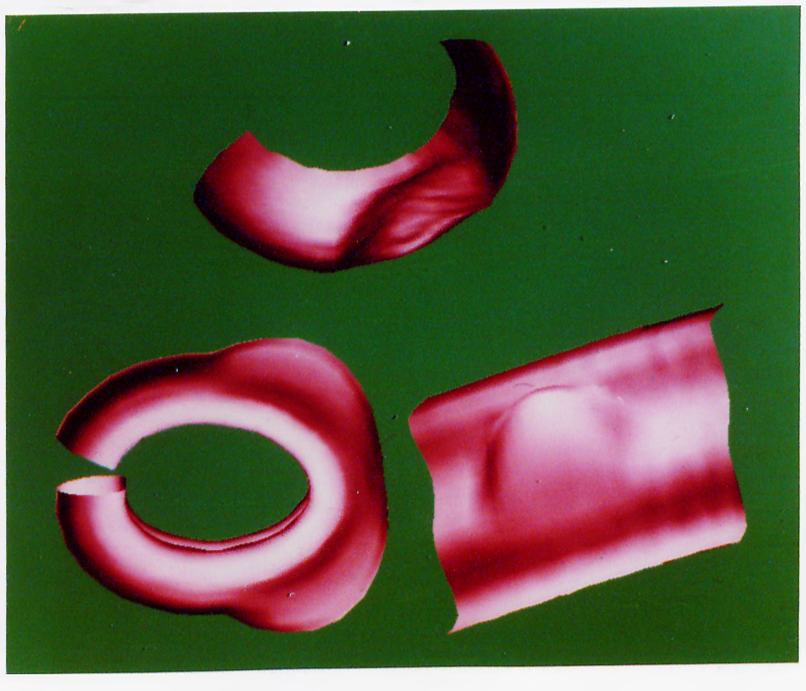


Figure 45 Some sweep objects deformed locally. Each picture was computed by Tb in about 5 seconds after the initial design of the figures.

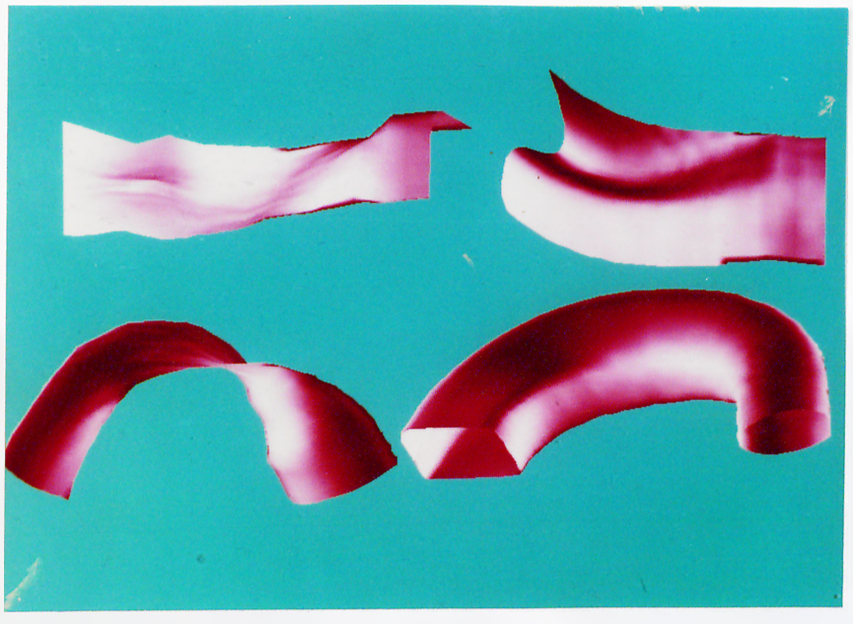


Figure 46 Some sweep objects with varying contours. Each picture was computed by Tb in about 6 seconds after the initial design of the figures.

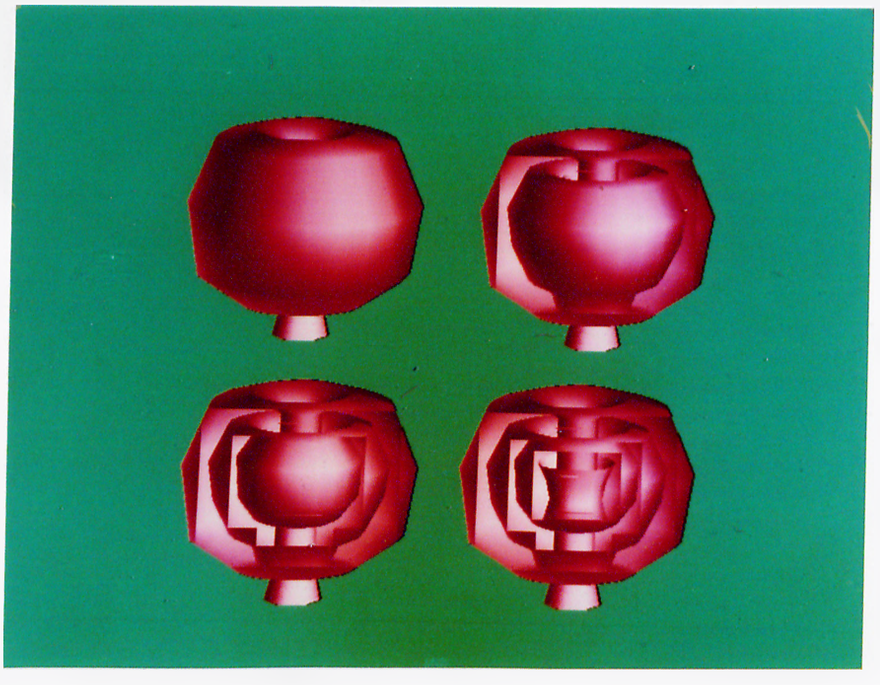


Figure 47 Transparent windows on a sweep object to inspect sublayers of the object. Each layer of the object can be inspected in 2 seconds.

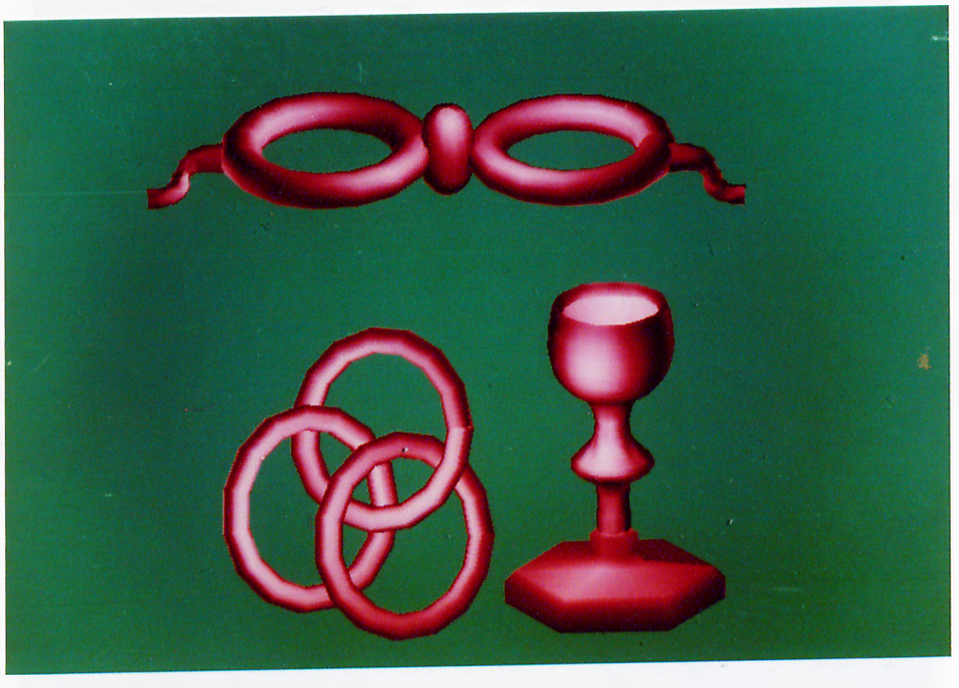


Figure 48 Some objects produced by composing the sweep objects. Each picture was computed by Tb in about 7 seconds after the initial design of the figures.

Figure 49 Some familiar objects produced by composing sweep objects. Each picture was computed by Tb in about 6 seconds after the initial design of the figures.

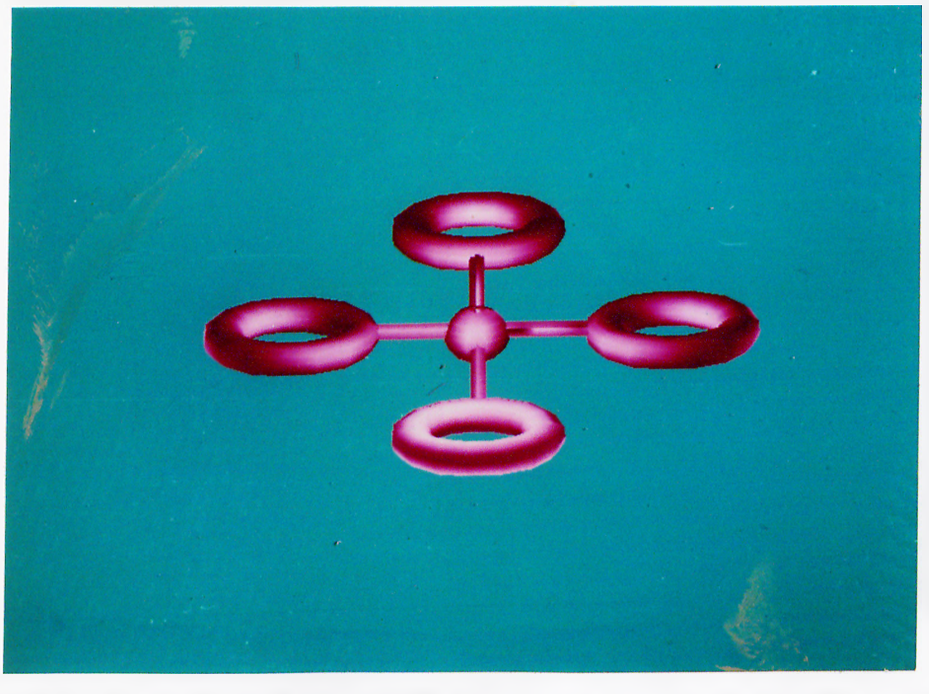


Figure 50 A punctured surface of genus 4. The picture was computed by Tb in about 10 seconds after the initial design of the figure.

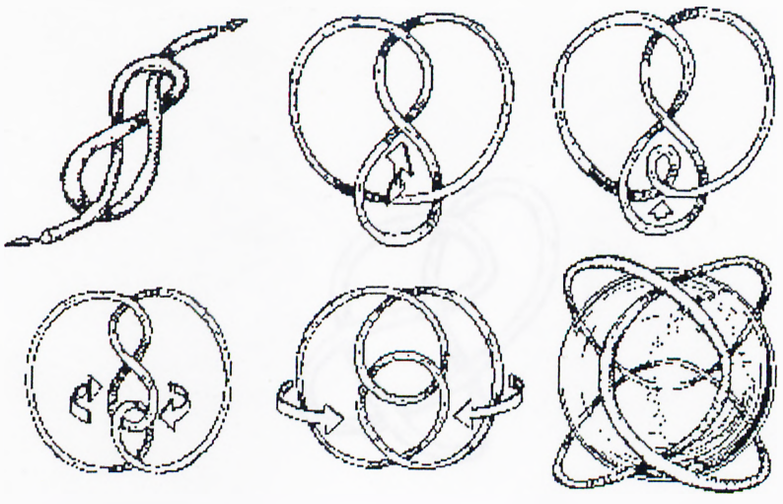


Figure 51 Projections of a knot (digitized from [25, p. 150]).

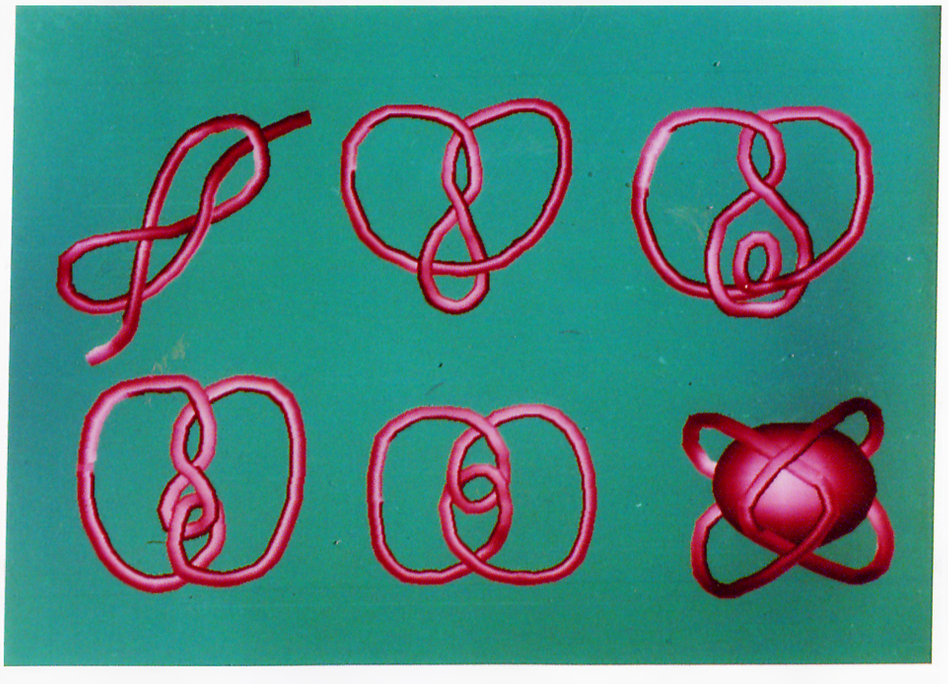


Figure 52 Projections of a knot. The picture was computed by Tb in about 40 seconds after the initial design of the figure.

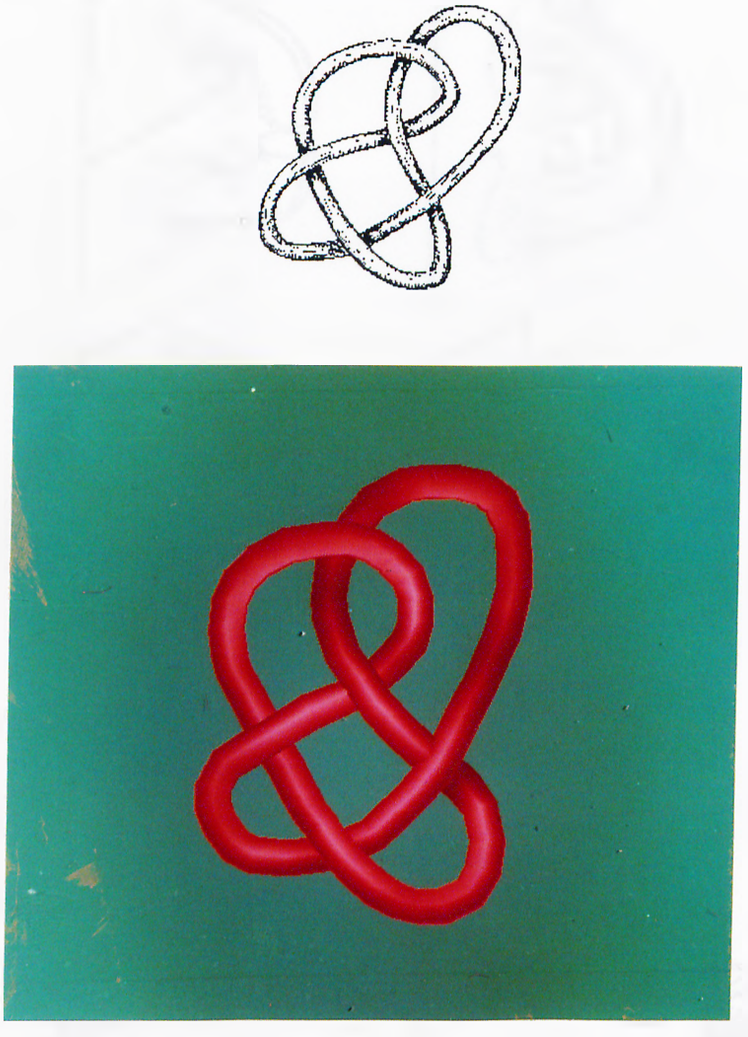


Figure 53 A knot (the first figure digitized from [25, p. 38] and the second produced by Tb in 6 seconds after the initial design of the figure).

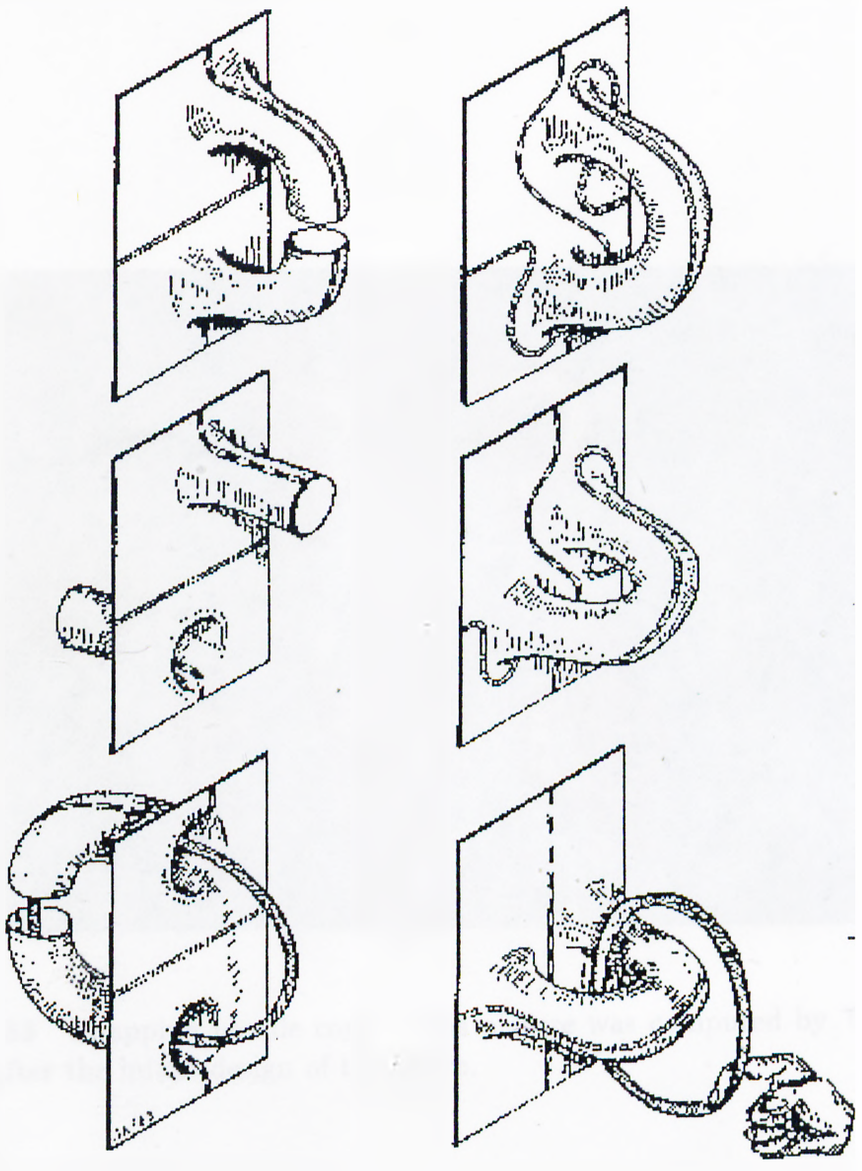


Figure 54 Swapping handle cores (digitized from [25, p. 140]).

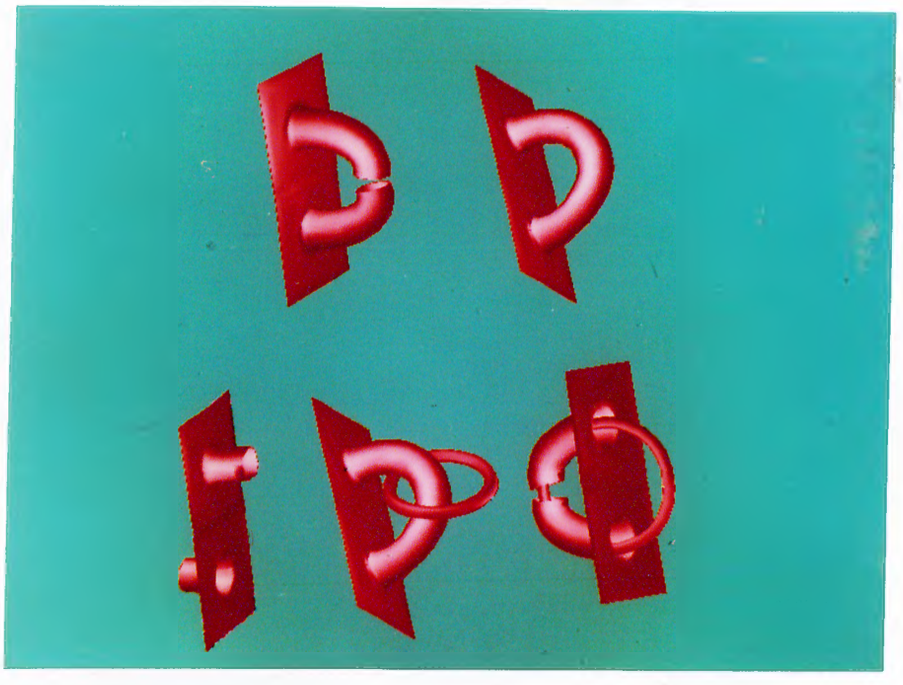
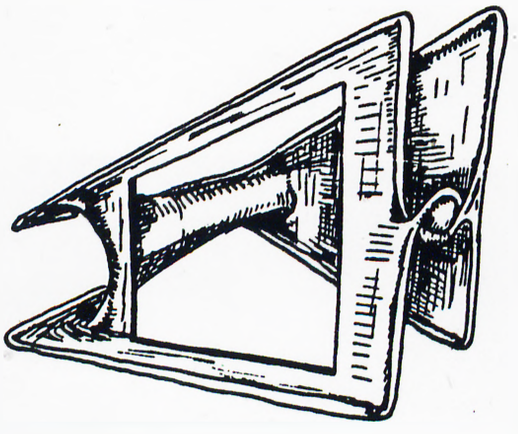


Figure 55 Swapping handle cores. This picture was computed by Tb in 1 minute after the initial design of the figure.



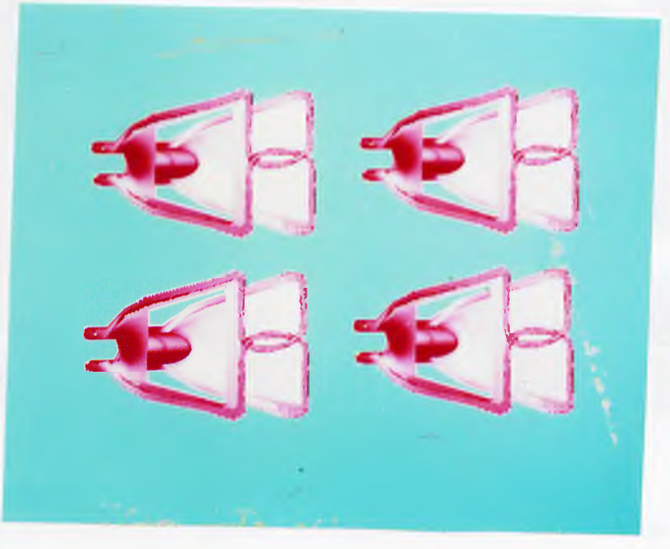
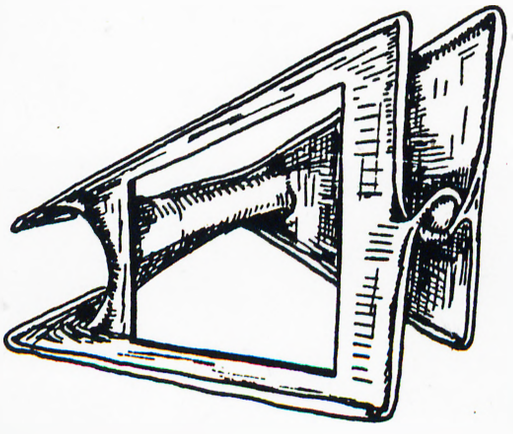


Figure 56 Eight knot (the first figure taken from [25, p. 37] and the second produced by Tb in 15 seconds after the initial design of the figure).



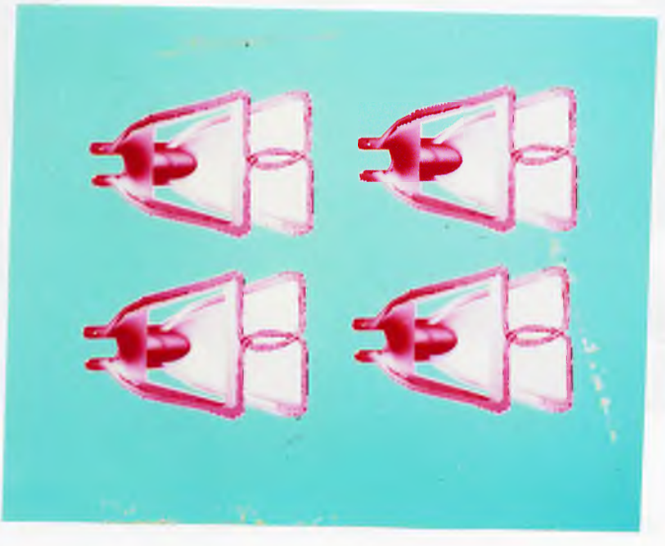


Figure 56 Eight knot (the first figure taken from [25, p. 37] and the second produced by Tb in 15 seconds after the initial design of the figure).

Figure 57 A picture of a statue rendered from wood (taken from [27, p. 87]).

References

[1] **Abelson, H., and diSessa, A.,** *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*, MIT Press (1982).

[2] **Akman, V.,** *Geometry and Graphics Applied to Robotics,* Technical Report No. CS–R8727, Center for Mathematics and Computer Science, Amsterdam (1987). Also in **Earnshaw, R. A. (ed.),** *Theoretical Foundations of Computer Graphics and CAD*, NATO ASI Series, Vol. F40, Springer-Verlag, pp. 619–638 (1988).

[3] **Akman, V.,** *Steps into a Geometer's Workbench,* Technical Report No. CS–R8726, Center for Mathematics and Computer Science, Amsterdam (1987). Also in **van der Burgh, A. H. P., and Mattheij, R. M. M. (eds.),** *Proceedings of First International Conference on Industrial and Applied Mathematics (ICIAM'87),* CWI Tracts, Vol. 36, Center for Mathematics and Computer Science, Amsterdam, pp. 257–275 (1987).

[4] **Akman, V.,** *Unobstructed Shortest Paths in Polyhedral Environments,* Lecture Notes in Computer Science, Vol. 251, Springer-Verlag (1987).

[5] **Akman, V., and Arslan, A.,** 'Sweeping with all graphical ingredients in a topological picturebook', *Computers & Graphics*, Vol. 16, No. 3 (1992).

[6] **Akman, V., Arslan, A., and Franklin, W. R.,** 'Implementing a topological picturebook', *Proceedings of 13th IMACS World Congress on Computation and Applied Mathematics,* Dublin (1991).

[7] **Akman, V., Arslan, A., and Franklin, W. R.,** 'Excursions to a topological picturebook', in **Santo, H. P. (ed.),** *Proceedings of First International Conference on Computational Graphics and Visualization Techniques (Compugraphics'91),* Vol. 2, Sesimbra, Portugal, pp. 344–361 (1991).

[8] **Arslan, A.,** *An Electronic Topological Picturebook,* Ph.D. Proposal, Department of Computer Engineering and Information Science, Bilkent University, Ankara (1990).

[9] **Arslan, A., and Akman, V.**, 'An electronic topological picturebook', Paper presented in *NATO Advanced Study Institute on Cognitive and Linguistic Aspects of Geographic Space*, Las Navas del Marques, Spain (1990).

[10] **Arslan, A., and Akman, V.**, 'Sweeping with all graphical ingredients', in **Baray, M., and Özgüç, B. (eds.)**, *Proceedings of Sixth International Symposium on Computer and Information Sciences (ISCIS VI)*, Vol. 2, Elsevier, pp. 1225–1234 (1991).

[11] **Arslan, A., İşler, V., and Akman, V.**, 'A procedure to sweep arbitrary curves', in **Harmancı, A. E., and Gelenbe, E. (eds.)**, *Proceedings of Fifth International Symposium on Computer and Information Sciences (ISCIS V)*, Vol. 2, İTÜ, pp. 895–904 (1990).

[12] **Barsky, B. A.**, 'A description and evaluation of various 3-D models', *IEEE Computer Graphics and Applications*, Vol. 4, No. 1, pp. 38–52 (1984).

[13] **Barsky, B. A.**, *Computer Graphics and Geometric Modeling Using Beta-Splines*, Springer-Verlag (1987).

[14] **Bartels, R. H., Beatty, J. C., and Barsky, B. A.**, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*, Morgan Kaufmann (1988).

[15] **Baumgart, B. G.**, *Geomed: Geometric Editor*, Technical Report No. STAN-CS-74-414, Computer Science Department, Stanford University, Stanford, CA (1974).

[16] **Bronsvoort, W. F., and Klok, F.**, 'Ray tracing generalized cylinders', *ACM Transaction on Graphics*, Vol. 4, No. 4, pp. 291–303 (1985).

[17] **Bronsvoort, W. F., Klok, F., and Post, F. H.**, *PLAMO: An Interaction Oriented Solid Modeling System with Shaded Images*, Technical Report No. 85-25, Deparment of Mathematics and Informatics, Delft University of Technology, Delft, Holland (1985).

[18] **Bronsvoort, W. F., Nieuwenhuizen, P. R. V., and Post, F. H.**, 'Display of profiled sweep objects', *Visual Computer*, Vol. 5, No. 3, pp. 147–157 (1989).

[19] **Cardelli, L.,** *Building User Interfaces by Direct Manipulation,* Technical Report No. 22, Systems Research Center, Digital Equipment Corporation, Palo Alto, CA (1987).

[20] **Choi, B. K., and Lee, C. S.,** 'Sweep surfaces modeling via coordinate transformations and blending', *Computer-Aided Design,* Vol. 22, No. 2, pp. 87–96 (1990).

[21] **Coquillart, S.,** 'A control-point-based sweeping technique', *IEEE Computer Graphics and Applications,* Vol. 7, No. 10, pp. 36–45 (1987).

[22] **Crowell, R. H., and Fox, R. H.,** *Introduction to Knot Theory,* Springer-Verlag (1977).

[23] **Fletcher, Y., and McAllister, D. F.,** 'A tension-compatible patch for shape-preserving surface interpolation', *IEEE Computer Graphics and Applications,* Vol. 9, No. 3, pp. 45–54 (1989).

[24] **Forsey, D. R., and Bartels, R. H.,** 'Hierarchical B-spline refinement', *ACM Computer Graphics,* Vol. 22, No. 4, pp. 205–211 (1988).

[25] **Francis, G. K.,** *A Topological Picturebook,* Springer-Verlag (1987).

[26] **Green, M.,** 'A survey of three dialogue models', *ACM Transactions on Graphics,* Vol. 5, No. 3, pp. 244–275 (1986).

[27] **Griffiths, H. B.,** *Surfaces,* Cambridge University Press (1981).

[28] **Gosling, T. H.,** 'The DUCT system of design for practical objects', *Proceedings of the Theory of Mechanisms on CAD in Mechanical Engineering,* Milan, pp. 305–316 (1986).

[29] **Güdükbay, U., and Özgüç, B.,** 'Free-form solid modeling using deformations', *Computers & Graphics,* Vol. 14, Nos. 3–4, pp. 491–500 (1990).

[30] **Hartman, E.,** 'Blending of implicit surfaces with functional splines', *Computer-Aided Design,* Vol. 22, No. 8, pp. 500–507 (1990).

[31] **Hearn, D., and Baker, M. P.,** *Computer Graphics,* Prentice-Hall, pp. 289–291 (1986).

[32] **Hoffmann, C., and Hopcroft, J.,** 'Automatic surface generation in computer-aided design', *Visual Computer,* Vol. 1, No. 2, pp. 92–100 (1985).

[33] **Hoffmann, C., and Hopcroft, J.**, 'The geometry of projective blending surfaces', *Artificial Intelligence*, Vol. 37, Nos. 1–3, pp. 357–376 (1988).

[34] **Huitric, H., and Nahas, M.**, 'B-spline surfaces: A tool for computer painting', *IEEE Computer Graphics and Applications*, Vol. 5, No. 5, pp. 39–47 (1985).

[35] **Karsenty, S.**, 'La construction d'interfaces utilisateurs', *Genie Logiciel & Systémes Experts*, No. 24, pp. 18–27 (1991).

[36] **Karsenty, S., Landay, J. A., and Weikart, C.**, *Inferring Graphical Constraints with Rockit*, Technical Report No. 17, Paris Research Laboratory, Digital Equipment Corporation, Rueil-Malmaison Cedex, France (1992).

[37] **Kauffman, L. H.**, *On Knots*, Princeton University Press (1987).

[38] **Kelley, A., and Pohl, I.**, *A Book on C*, Benjamin/Cummings (1984).

[39] **Kernighan, B. W., and Ritchie, D. M.**, *The C Programming Language*, Prentice-Hall (1978).

[40] **Klok, F.**, 'Two moving coordinate frames for sweeping along a 3-D trajectory', *Computer Aided Geometric Design*, Vol. 3, pp. 217–229 (1986).

[41] **Kneale, D.**, 'Shaping ideas: A topologist shows the world of math by seeing the unseen', *Wall Street Journal* (March 18, 1983).

[42] **Löwgren, J.**, *'History, state, and future of user interface management systems'*, *SIGCHI Bulletin*, Vol. 20, No. 1 (1988).

[43] **Markowsky, G., and Wesley, M. A.**, 'Generation of solid models from two-dimensional and three-dimensional data', in **Pickett, M. S., and Boyse, J. M. (eds.)**, *Solid Modeling by Computer: From Theory to Application*, Plenum, pp. 23–51 (1986).

[44] **Martin, R. R., and Stepson, P. C.**, 'Sweeping of three-dimensional objects', *Computer-Aided Design*, Vol. 22, No. 4, pp. 223–234 (1990).

[45] **Middleditch, A. E., and Sears, K. H.**, 'Blend surfaces for set-theoretic volume modeling systems', *ACM Computer Graphics*, Vol. 19, No. 3, pp. 161–170 (1985).

[46] **Myers, B. A.,** *Creating User Interfaces by Demonstration,* Academic Press (1988).

[47] **Newman, W. M., and Sproull, R. F.,** *Principles of Interactive Computer Graphics,* McGraw-Hill, pp. 398–401 (1981).

[48] **Pentland, A. P.,** *Perceptual Organization and the Representation of Natural Form,* Technical Report No. TN–357, Artificial Intelligence Center, SRI International, Menlo Park, CA (1985).

[49] **Pentland, A. P.,** *Supersketch^{TM} User Manual,* Artificial Intelligence Center, SRI International, Menlo Park, CA (1985).

[50] **Post, F. H., and Klok, F.,** 'Deformations of sweep objects in solid modeling', in **Requicha, A. A. G. (ed.),** *Eurographics'86 Proceedings,* North-Holland, pp. 103–114 (1986).

[51] **Reewes, W. T., Ostby, E. F., and Leffler S. J.,** 'The Menv Modelling and Animation Environment', *Journal of Visualization and Computer Animation,* Vol. 1, No. 1, pp. 33–40 (1990).

[52] **Requicha, A. A. G.,** 'Representations for rigid solids: Theory, methods, and systems', *ACM Computing Surveys,* Vol. 12, No. 4, pp. 437–464 (1980).

[53] **Ringel, G.,** *Map Color Theorem,* Springer-Verlag (1974).

[54] **Rogers, D. F., and Adams, J. A.,** *Mathematical Elements for Computer Graphics,* McGraw-Hill, pp. 394–400 (1989).

[55] **Roth, S. D.,** 'Ray casting for modeling solids', *Computer Graphics and Image Processing,* Vol. 18, No. 2, pp. 109–144 (1982).

[56] **Stillwell, J.,** *Classical Topology and Combinatorial Group Theory,* Springer-Verlag (1980).

[57] **Wang, W. P., and Wang, K. K.,** 'Geometric modeling for swept volume of moving solids', *IEEE Computer Graphics and Applications,* Vol. 6, No. 12, pp. 8–17 (1986).

[58] **Woodward, C. D.,** 'Methods for cross-sectional design of B-spline surfaces', in **Requicha, A. A. G. (ed.),** *Eurographics'86 Proceedings,* North-Holland, pp. 129–142 (1986).

[59] **Woodward, C. D.**, 'Skinning techniques for interactive B-spline surface interpolation', *Computer-Aided Design*, Vol. 20, No. 8, pp. 441–451 (1988).

[60] **Woodwark, J. R.**, *Computing Shape*, Butterworths (1986).

A Technical Details of the Tb Package

Tb was written in the *C* programming language [38, 39]. Its user interface system is based on the Sunview windowing system. It runs on a color (or black and white) Sun workstation. Its source code can be found in the directory */home/usr3/arslan/Tb* on the Sun computer system at Bilkent University. Tb consists of the following source files:

- animation.c creates all the interaction items for animation and includes some functions and procedures to animate a sweep object.
- blending1.c, blending2.c include blending functions and procedures. In addition, each interaction item for blending is created by the help of these.
- coloring.c includes the operations available in the menu COLORING.
- colormap.c builds a colormap, creates different colormaps, and selects the background color.
- edit.c includes editing functions and procedures and creates interaction items for editing. The contour, the trajectory, the profile, and the depth-modulation curves can be edited by the help of this program.
- grey_level.c provides a dump of the color screen as Postscript code. (This option is only available for a color screen.)
- grid.c determines the number of grids on the produced sweep object.
- hole.c makes it possible to create a polygonal hole on a produced sweep object.
- image.c includes the image scaling and translation procedures. Some procedures to provide the interaction between the user and Tb are also in this file.

- include_file.h declares some global structures and variables. This file is included by all the files of Tb.

- knots.c helps the user to create a knot or to edit an existing knot. In addition, some procedures providing the interaction can be found in this file.

- local_def.c includes code to obtain deformations on a produced sweep object.

- main.c the main program; Tb starts its execution with this program.

- main_canvas.c creates the main drawing canvas; each event on the canvas is directed by this program.

- makefile automates the compilation of Tb. (In order to build an executable copy of Tb from scratch, just type: *make*.)

- modulated_sweep.c includes some functions and procedures to obtain a depth-modulated sweep object. Code to provide the interaction between the user and Tb is also included.

- nonprofiled_sweep.c includes some functions and procedures to obtain a nonprofiled sweep object. Code to provide the interaction between the user and Tb is also included.

- panel1.c, panel2.c include facilities to design general menus, panels, and interaction systems.

- pro_mod_sweep.c includes some functions and procedures to obtain a profiled and depth-modulated sweep object. Code to provide the interaction between the user and Tb is also included.

- profiled_sweep.c includes some functions and procedures to obtain a profiled sweep object. Code to provide the interaction between the user and Tb is also included.

- save_load.c saves or loads the data for an object or a selected area of an image. The interactors for saving and loading are also created.

- shade_polygon.c includes a constant shading technique, a wireframe output option, and some procedures and functions to display auxiliary curves of a produced sweep object.

- shading.c includes the Gouraud shading method, the z-buffer method to sort each point on the object, and the procedures for the calculation of the surface normals and colors.
- skinning.c includes code to produce sweep objects with varying contours.
- text.c creates some bitmap characters, assign them to the any key of the keyboard, and displays them.
- topology.c normalizes the given symbols which represent a polyhedron and displays the obtained handles (in symbolic form) and a sphere with these handles on it.
- transformations.c performs 3-D rotation, scaling, and translation and creates some interactors.
- transparent_shading.c includes a shading technique to inspect the layers of an object, the structure of the shading, and some auxiliary procedure and functions for shading.
- twisting.c performs twisting and creates some interactors.

These files consist of a total of about 31000 lines. The compilation of Tb takes about 7 minutes on a Sun-4. The size of the compiled code is 1.048 Mbytes.

Tb uses 8 directories in the execution time.

- Draster is used to save images in the raster format. The image files in this directory can be loaded by a dynamic menu in Tb. The user must select the menu LOAD and then, the submenu LOAD RASTER.
- Dnormal includes normal files which consist of vertex coordinates of the auxiliary curves and some information about objects. The files in this directory can be also loaded by a dynamic menu in Tb. The user must select the menu LOAD and then, the submenu LOAD NORMAL.
- Dhandles includes some data files which are used to draw handles on a sphere.
- Dcursors includes some cursor image files which are used to display different cursors in Tb.

- Dhexa is used to save the image files dumped as hexadecimal numbers. The files in this directory include each four points of the screen as a hexadecimal number.
- Dchar includes the bit-mapped character files produced by the help of the menu TEXT in Tb.
- Dpostscript is used to save the image files dumped as Postscript code.
- Dauxfiles includes some data files prepared by the user. For example, if it is necessary to enter coordinates of the auxiliary curves from a data file to produce a sweep object with varying contour, the user must use this directory for those files which are to be loaded.